electronics

MDPI

Review

Methods of Intelligent Control in Mechatronics and Robotic Engineering: A Survey

Iuliia Zaitceva [1,2] and Boris Andrievsky [1,2,*]

[1] Control of Complex Systems Lab., Institute of Problems in Mechanical Engineering of Russian Academy of Sciences (IPME RAS), 61 Bol'Shoy Pr. V.O., 199178 Saint Petersburg, Russia
[2] Department of Theoretical Cybernetics, Faculty of Mathematics and Mechanics, Saint-Petersburg State University, Stary Peterhof, Universitetsky Prospekt, 198504 Saint Petersburg, Russia
* Correspondence: boris.andrievsky@gmail.com, Tel.: +7-812-321-4776

Abstract: Artificial intelligence is becoming an increasingly popular tool in more and more areas of technology. New challenges in control systems design and application are related to increased productivity, control flexibility, and processing of big data. Some kinds of systems require autonomy in real-time decision-making, while the other ones may serve as an essential factor in human-robot interaction and human influences on system performance. Naturally, the complex tasks of controlling technical systems require new modern solutions, but there remains an inextricable link between control theory and artificial intelligence. The first part of the present survey is devoted to the main intelligent control methods in technical systems. Among them, modern methods of adaptive and optimal control, fuzzy logic, and machine learning are considered. In its second part, the crucial achievements in intelligent control applications in robotic and mechatronic systems over the past decade are considered. The references are structured according to the type of such common control problems as stabilization, controller tuning, identification, parametric optimization, iterative learning, and prediction. In the conclusion, the main problems and tendencies toward intelligent control methods improvement are outlined.

Keywords: cyber-physical systems; artificial intelligent; control; adaptation; tuning; learning; optimization; prediction; real-time; smart systems

check for updates

Citation: Zaitceva, I.; Andrievsky, B. Methods of Intelligent Control in Mechatronics and Robotic Engineering: A Survey. *Electronics* **2022**, *11*, 2443. https://doi.org/10.3390/electronics11152443

Academic Editors: Ji-Hyeong Han, Inbae Jeong and Ki-baek Lee

Received: 8 July 2022
Accepted: 2 August 2022
Published: 5 August 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

From the automation viewpoint, the intelligent control strategy can be interpreted quite flexibly. Its concept differs from classical feedback one in that it is dynamically implemented in real-time while the system performs useful functions. Thus, classical adaptive control can be considered an intelligent one, if the control law is constantly updated. From the classification viewpoint, systems of such a kind can be classified as borderline [1]. The development of methods that originate from computational mathematics, then moving into the field of programming, processing and analyzing big data, and, recently, have been actively used in the engineering systems control, is well traced when analyzing the sources on the intelligent control approaches for mechatronic. The explosive growth of literature falls at the beginning of the 2010s and does not cease to decrease.

In robotics and mechatronics, various approaches and areas of artificial intelligence are used, such as machine learning, artificial neural networks (NN), fuzzy logic, evolutionary computing, and genetic algorithms. The field of machine learning includes methods such as classical learning (supervised and unsupervised), ensemble methods, reinforcement learning, neural networks, and deep learning. Intelligent control algorithms use various optimization, statistical and probabilistic methods to extract effective patterns from actions taken and analyze large data sets [2]. Classical learning and ensembles are mainly used for processing and classifying data sets versus NNs, and reinforcement learning that suitable for use in the automatic control field, planning and logistics [3–5]. At the same time, NNs are especially effective when working with complex data and unlabeled features.

An important issue is the reliability and safety of using intelligent control algorithms in autonomous systems interacting with the physical world. This issue is covered in the review article [6], where the non-asymptotic analysis (finite time) of the convergence of intelligent control algorithms is considered.

The present survey purpose is a description of the most used modern intelligent control methods and this application researches in the field of mechatronics, to identify trends and open issues in intelligent control.

The remainder of the paper is organized as follows. Various smart control methods are recalled in Section 2. Section 3 is devoted to intelligent methods applications in engineering control problems. Concluding remarks in Section 4 finalize the paper.

2. Smart Control Methods

Although intelligent control is a separate area, in order to apply new concepts, such as NN in a control loop, it is necessary to use various mathematical methods that underlie the classical automatic control theory. Thus, intelligent control is a combined method in which automatic control methods are actively used, while the system is able to learn from its previous experience, making it possible to improve its performance every time.

Let us consider the most widely used methods of modern control theory within the intellectual approach framework. Some of them are widely known and do not require a detailed description. Therefore, in the present paper, the most attention is paid to machine learning.

2.1. Adaptive Control Methods

Adaptive control, like optimal control, has a well-developed theoretical basis and a rigorous mathematical justification, cf. [7]. Within the framework of the classical theory of automatic control, this method has become the starting point for intelligent control, since the synthesis of adaptive controllers provides a given quality of system operation under conditions when the plant parameters and characteristics of the external environment are not known or are changing in an indefinite way. The principle of adaptation has evolved from self-tuning controllers to adaptive learning systems and can be used at the heart of intelligent control methods, cf. [8].

Algorithms for adaptive control of time-discrete systems have found their application in the field of artificial intelligence in the works of V. Yakubovich, who obtained several algorithms for linear classification model training, see [7,9–13]. In the recent paper [14], the "Stripe Algorithm" (SA) by V. Yakubovich, successfully used for machine learning, is considered from a modern point of view. By the numerical analysis, it is shown that the SA is well suited for online machine learning, providing higher performance than traditional linear learning methods allow. Lipkovich [14] presents various ways of reducing loss optimization problems to solving systems of inequalities, including both classification and regression problems. SA is compared in [14] with modern linear analogues, such as linear and logistic regression, and learning with gradient descent. It is noted that complex non-linear models are likely to outperform the SA, but the last one has the usual advantages of a linear model, such as ease of implementation, development, and explainability.

As is mentioned in [15], the existence of a theory does not guarantee its application in practice, especially in real conditions of construction and operation of control systems.

2.2. Optimization Methods

Historically, optimization methods appeared much earlier than machine learning and were used to find the extremum of the [16] function. Optimality theory formed the basis of most machine learning problems, which can be generally formulated as minimizing some functional J concerning some parameter θ: $J(\theta) \longrightarrow \min_{\theta \in \chi}$. The form of the value to be minimized depends on the machine learning method. So, for example, in the case of a classification or regression problem, they minimize the prediction error on the existing sample, and in the theory of reinforcement learning, they find the maximum benefit from

the actions of the plant. This goal can be achieved using any search algorithm. Thus, mathematical optimization has many types, methods, and applications.

2.3. Artificial Neural Networks

The history of the development of artificial neural networks experienced ups and downs, until the beginning of the 90 s when their theoretical foundation had been formed.

The artificial neuron structure is described by function $\mathbb{R}^n \to \mathbb{R}$ [17]. The artificial neuron has n inputs x_i with weights w_i, by which the signal passing through the connection is multiplied. After that, the weighted $x_i w_i$ signals are sent to the adder. The sum of weighted neuron outputs is found as: $net = \sum_{i=1}^{i=n} w_i x_i = w^{\mathrm{T}} x$. This weighted sum is then passed through the activation function $\varphi(net)$, which converts it into a number that is fed to the output neuron. A simple diagram of the operation of a neuron is shown in Figure 1, where the unit-step (Heaviside) function is used as the activation one.

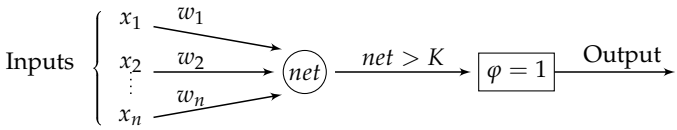


Figure 1. Neuron Structure

The task of NN learning is reduced to determining such an architecture and weights under which the learning error tends to a minimum for all possible training samples. The NN learning process consists of the search for a set of weights that transform the input signal into the desired output. Since the learning error function can have an arbitrary form, NN learning is generally described in terms of a multi-extremal non-convex optimization problem.

The use of NNs is efficient for nonlinear and noisy systems control. It provides adaptability, and after training, the NN can operate in real-time. The ongoing development of the structures and properties of NN is aimed at overcoming the existing shortcomings. The heuristic NN training approaches can lead to ambiguous solutions and deadlocks, and also require the preparation of a training sample. Significant disadvantages for NN applications in robotics are caused by the long time spent on training, increasing the nonuniqueness of training results prediction, the risk of improper control of expensive plants, and the present-day implementation of NN is currently possible in the form of ultra-large integrated circuits only.

2.4. Fuzzy Logic Approach

The fuzzy logic (FL) approach proposed by L. Zadeh is based on the concept of a fuzzy set as an object with an element membership function to the set, taking values on the interval $[0, 1]$, see [18,19]. It turned out to be possible to represent FL reasoning in the form of NN. To do this, the role of the activation function of the neuron should be performed by the membership function, the connections between neurons should be the signal transmission, and the logical t-norms and k-norms are presented as special types of neurons. At the moment, many neuro-fuzzy networks have been developed, which are approximately described by the universal form of [20] approximators. Their characteristic feature is the human-like style of reasoning and the connectionist structure of the NN.

2.5. Reinforcement Learning

In the concept of reinforcement learning, the notions of agent and environment are used. The agent performs action a_t in the state s_t and receives a reward r_t for this. Depending on the reward value, the agent performs the next action a_{t+1} in the state s_t, for which it receives the reward r_{t+1}. The agent, in the process of interacting with an uncertain dynamic environment, learns the so-called "optimal decision policy", which reflects the state of the agent after performing optimal actions [21–23]. Schematically, the reinforcement learning procedure is shown in Figure 2.

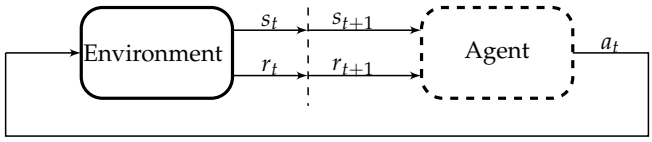


Figure 2. Reinforcement Learning Procedure.

Kurrek et al. [24] described a general approach to developing a control policy for various robots, environments, and manipulative tasks. The environment can be either physical or virtual, implemented as a computer program. Kim et al. [25] demonstrated the efficiency of reinforcement learning by the example of an inverted pendulum, controlled by a proportional-integral-derivative (PID) controller, implemented virtually. An important aspect is an interaction between the agent and the environment in the real-time mode, see [26]. The agent receives real-time observation of the reaction of the environment when random or suboptimal actions are taken, and an assessment of these actions in the form of a reward. An iterative reinforcement learning process usually consists of two steps: policy evaluation and subsequent policy improvement.

In the literature, the concept of reinforcement learning is referred to under several alternative names: approximate dynamic programming [27], neuro-dynamic programming [28], adaptive critic [29]. Werbos [30] used the broad term "approximate dynamic reinforcement learning programming [30]. Approximate dynamic programming in a broad sense includes all tools available from a computational point of view for finding the most accurate possible solution to the Bellman equation, as required for the solution of optimality associated with dynamic programming. From the point of view of control theory, reinforcement learning can be considered as an adaptive optimal control provided by an adaptive controller converging to the optimal [31].

The general discrete-time stochastic system in the reinforcement learning problem, or a feedback control problem, is usually expressed as a one-step Markov model in which the fully observable system $s_{t+1} = F(s_t, a_t, w_t)$ responds on the action a_t. Random variable w_t represents the uncertainty or perturbation. The learning goal at each time t is to find a policy π indicating what decision should be made given the state in order to minimize (or maximize) the sum of all penalties (or rewards):

$$\min_{\pi(\cdot)} \mathbb{E}\left[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} R(s_\tau, u_\tau = \pi(s_\tau)) \mid s_t\right], \tag{1}$$

where $0 < \gamma \leqslant 1$ is the discount factor, R is the reward function, $\mathbb{E}[\cdot]$ denotes the conditional expectation operator.

The mathematical formulation of such a sequential decision problem is called the Markov Decision Process (MDP) [32]. The formulated MDP problem is solved precisely with the help of dynamic programming, in which the value (cost) function, estimating the sum of all future costs under the optimal decision-making policy, is introduced in the following time-recursive form (the *optimal Bellman equation*):

$$V^*(s_t) = \min_{\pi(\cdot)} \{R(s_t, \pi(s_t)) + \gamma \mathbb{E}[V^*(s_{t+1}) \mid s_t, \pi(s_t)]\}. \tag{2}$$

The optimal policy, corresponding (2) is obtained as the solution to the following optimization problem:

$$\pi^*(s_t) = \operatorname{argmin}_{\pi(\cdot)} \{R(s_t, \pi(s_t)) + \gamma \mathbb{E}[V^*(s_{t+1}) \mid s_t, \pi(s_t)]\}. \tag{3}$$

The exact solution of (2) cannot be obtained for most practical problems due to the computational complexity associated with the huge size of the state space and actions, which is called the "curse of dimensionality". Approximate Dynamic Programming Reinforcement Learning (ADPRL) was suggested as a practical method for developing a decision-making policy aimed to achieve the online optimal feedback policy.

The Bellman equation is replaced by a recursion one, and the value function is iteratively updated depending on the prediction error (the temporal difference). A certain approximation is used to represent a value function or action policy, such as a parametric model, NN approximation, nearest neighbor averaging, Taylor series, etc. The main idea of approximate dynamic programming is to represent the quality function. However, due to the random choice of initial weights, it needs to learn to converge many times. The solution to this problem is proposed in [33], where an equivalent is constructed between the initial action of the NN and the proportional-integral-derivative (PID) controller, as a result of which the PID controller can control the choice of initial weights.

If the system model dynamics are unknown, then the so-called Q-learning is used, in which the following Q-function is introduced with the state-action argument [34]:

$$Q^*(s_t, a_t) = R(s_t, a_t) + \gamma \min_{a'} \mathbb{E}[Q^*(s_{t+1}, a_t)], \tag{4}$$

$$\pi^*(s_t, a_t) = arg \min_{a(\cdot)} Q^*(s_t, a_t), \tag{5}$$

where Q^* is the optimal state-value function, that corresponds to the optimal strategy π^*.

An important question in learning is what decision an agent must make in order to quickly learn new information that determines a policy of action. Q-function (4) converges only when all actions have a non-zero probability of being chosen by the decision policy, i.e., from a learning point of view, decisions should be made in order to gain more valuable information about the model and uncertainty. On the other hand, in order to achieve good results, you should choose the best option based on current knowledge.

Probabilistic deduction for learning is considered an efficient approach where the problem of Bayesian estimation with Gaussian process regression is posed and produces a fully deterministic approximate inference for policy estimation, making it computationally efficient [35]. This approach requires a task-specific scenario. If the agent is given a new goal, then the model needs to be retrained. In [35], the probabilistic learning inference is extended for the case of linear feedback controller tuning. To do this, an improved policy is added to the Bellman equation:

$$\pi(s) = \text{argmax}_{a \in A(s)} \int P^a_{s,s'}, [R^a_{s,s'} + \gamma V(s')] ds', \tag{6}$$

where $P^a_{s,s'}$ is the probability of transition to state $s^{'} = s_{t+1}$ when performing action a, $V(s')$ is a function values. The proposed learning structure was tested in [35] on a quadrocopter control system with uncertain dynamics and showed the convergence of learning depending on the number of iterations.

In [36], for solving the classical optimal control problem and reinforcement learning, a method of dynamic programming of Gaussian processes and a reinforcement learning algorithm based on an approximate value function are employed. The expected value in [36] is proposed to be calculated as:

$$U(\tilde{x}) = \rho \mathbb{E}_V[V^*_k(\tilde{x})|X_k] + \frac{\beta}{2} \log(\text{var}_V[V^*_k(\tilde{x})|X_k]), \tag{7}$$

where $\tilde{x} \in \tilde{X}_k$, $\tilde{x}$ is the state vector, ρ, β are the weight coefficients, E_V is the normal distribution, var_V denotes the variance. The state-value function V^*_k is the argument of the expectation and covariance functions.

3. Application Intelligent Methods in Engineering Control Problems

3.1. Stabilization and Program Control Problems

The tasks of stabilization and program control imply the presence of feedback in the loop. Usually, the system state vector is not available for measurement, so the control strategy is based on the available measured output. Standard tasks in mechatronics and robotics are tasks such as speed stabilization and trajectory tracking. These variables are

easy to measure at the system output. Ref. [37] represents a machine learning approach for a quadrocopter. In this paper was proposed the policy π_θ is parameterized by θ and is differentiable for parameters. The objective functional $J(\pi_\theta)$ is differentiable to θ, i.e., optimized by the gradient method (policy gradient method). For this, it is used the average reward gradient formula for the strategy parameters and their stochastic estimates. The most common kind of gradient estimator is written in the following form:

$$\hat{g} = \hat{\mathbb{E}}_t[\nabla_\theta \log_{\pi\theta}(a_t|s_t)\hat{A}_t], \tag{8}$$

where $\hat{A}_(s_t, a_t) = Q(s_t, a_t) - \hat{V}(s_t)$ is the value of the advantage function at time step t, $\hat{\mathbb{E}}_t$ is the empirical average over a finite set of samples, obtained by alternating the processes of sample generation and π_θ policy improvement. In this reinforcement learning problem, the dynamic model is unknown, possibly non-differentiable, and needs to be trained. So, it will increase the variance of the gradient estimates. In this connection, the authors have developed a more reliable method for optimizing the policy by gradually expanding the agent performance. For this, the following objective function is proposed, after differentiation of which (8) is obtained:

$$J(\theta) = \hat{\mathbb{E}}_t[\min(r(\theta)\hat{A}_t, \mathrm{clip}(r(\theta), 1-\varepsilon, 1+\varepsilon))], \tag{9}$$

where ε is a hyperparameter, $r(\theta) = \pi_\theta / \pi_{\theta old}$.

Gradient $\hat{g}$ in (8) is obtained by differentiation of (9). The reward function is chosen as simply as possible in the form:

$$r_t(e_{xt}, e_{yt}, e_{zt}) = a - \sqrt{e_{xt}^2 + e_{yt}^2 + e_{zt}^2}, \tag{10}$$

where e_{xt}, e_{yt}, e_{zt} are the quadcopter position mismatches on the corresponding coordinate axis; constant a is introduced to ensure that the quadcopter receives a reward in the limited flight area.

In [38], entropy-adaptive reinforcement learning for walking a lightning-fast tripod robot with a soft membrane vibration drive is considered. For this problem type, controller development with feedback and data collection for learning is required. So, the authors use a model-free learning method. The Controller is defined as a Gaussian normal distribution policy: $f_\varphi(s_t) = (\mu_t, \sigma_t)$, where μ_t and σ_t are the mean and standard deviation, and φ is the controller parameter. The function f_φ is given as NN. Action strategies about starting as $N(a_t; f_\varphi(s_t))$. Reward function is chosen as:

$$r(s_t) = -d_t - \delta\theta_t + c, \tag{11}$$

where $\delta\theta_t$ is the angular difference between the current and desired position, d_t is the root mean square error between the robot's current position and the final position, and c is the coefficient. Optimal action policy to Maximize Entropy Solution is as:

$$\pi_\alpha^* = \arg\max_\pi \mathbb{E}_{\tau\sim P,\pi}\Big[\sum_{t=0}^{\infty} \gamma^t(\hat{r}(s_t, a_t) + \alpha H(\pi(\cdot|s_t)))\Big], \tag{12}$$

$$H(\pi_\varphi(\cdot|s)) = \mathbb{E}_{\alpha\sim\pi_\varphi}[-\log(\pi_\varphi(a|s))] \tag{13}$$

where $\hat{r}(s, a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}[r(s')]$, entropy temperature $\alpha \in [0, \infty)$. According to the adaptive soft actor-critic algorithm, it is necessary to minimize the value functional using the stochastic gradient descent method.

The reinforcement learning approach described above is not applicable when the control goal changes over time. A solution to this problem is proposed in [39], where one manages a state-action-reward set to learn to simulate a constant goal in each set. This approach was tested on a speed control system on the model and the experimental vehicle with different setpoints.

An important approach in building intelligent systems is employing the Bayesian statistical methods [40]. Recent data from human brain studies led to the hypothesis whereby the brain uses Bayesian internal models during certain types of sensorimotor learning to optimize the performance of a particular task. The resulting activity of certain neural ensembles can be modelled as cooperative Bayesian processes. These neural signal processing concepts can be applied in applications ranging from rehabilitation engineering to artificial intelligence.

Endo [41] proposed a new approach for predictive control of a robot in a partially observable environment using machine learning methods consisting of several steps. First, the episodic memory E is calculated, consisting of the time sequence n of events e and the goal g that the robot pursued during each episode: $E = \{(e_1, e_2, ..., e_n), g\}$. The episodes are separated based on targets, which are used as keys to retrieve the corresponding episodes from memory. The event consists of observing $o = \{z_1, z_2, ...z_m\}$ received from m-measurement probes, actions performed by motor $b = \{\sigma_1, \sigma_2, .., \sigma_\beta\}$ and r rewards: $e = \{o, b, r\}$. The goal of the robot g is given by the sequence $\{z_1^g, z_2^g, ..., z_m^g\}$. The anticipatory behaviour of the robot is computed using event sampling, episode recall, event matching, and behaviour selection. At each stage, different machine learning is used: temporal difference learning (TDL), learning by example, recursive Bayesian estimation, and MDP. The utility function of the i-th event is proposed as:

$$U(e_i) = r_i - \sum_{e' \in E} p(e' \mid b_{i+1}, e_i) U(e'), \tag{14}$$

where p is the transition probability. The best action strategy is written as:

$$b^* = \arg\max_b \frac{1}{|\Gamma^+(b)|} \sum_{E \in \Gamma^+(b)} \sum_{e' \in E} p(e' | b, \hat{e}_{[E]}) U(e'), \tag{15}$$

where function $\Gamma^+(b)$ returns a subset of episodes containing matched events.

Intelligent power generation is rapidly developing and attracting the attention of researchers. Classical methods are becoming insufficient in this area due to the growing scale of systems and various uncertainties. Sun and You Sun and You [42] proposed an alternative to these methods in the burgeoning era of machine learning and data-driven governance. In [42], an overview of smart methods used in the energy sector that can be applied in monitoring, control, optimization, and fault detection of power generation systems. Control tasks with stochastic external influences are difficult to solve by classical methods and have complex control structures that do not always satisfy the quality of work [43,44]. These devices include wind turbines that convert wind energy into mechanical energy. For them, intelligent control methods are especially relevant and powerful. Sierra-Garcia et al. [44] investigate the wind turbine pitch control in which there is dependence on the nonlinear dynamics of the wind turbine, the relationship of variables, and environmental uncertainties. A speed-improved hybrid learning is proposed to solve this problem, consisting of a PID controller and a learning observer. Such a control structure turned out to be more efficient than a fuzzy controller and NN and allows to reduce the output power error by 41%. The control law is described by the following expressions:

$$s_t = f_{se}(P_{err}(t_i), P_{err}(t_{i-1}), v_w(t_{i-1})), \tag{16}$$

$$T_{s_{t-1},a_{t-1}}(t_i) = f_{pu}(r_t, T_{s_{t-1},a_{t-1}}(t_{i-1}) \tag{17}$$

$$\theta_R(t_i) = f_{as}(T_{s_t}(t_i), \varepsilon(t_i), eW(t_i))), \tag{18}$$

$$\theta_{ref}(t_i) = \theta_R(t_i) - \theta_{PID}(t_i) \tag{19}$$

$$[\varepsilon(t_i), eW(t_i)] = \begin{cases} f_{LO}(P_{err}(t_i), P_{err}(t_{i-1})...P_{err}(t_{i-N})) \\ [\varepsilon(t_{i-1}), eW(t_{i-1})], \end{cases} \tag{20}$$

$$[T(t_i), bT(t_i)] = \begin{cases} f_{LOT}(P_{err}(t_i), P_{err}(t_{i-1})...P_{err}(t_{i-N})) \\ [T(t_{i-1}), bT(t_{i-1})]. \end{cases} \tag{21}$$

where P_{err} is the error between the given and output power, v_w denotes the wind speed, f_{se} is the state estimator function, f_{pu} is the policy update function, f_{as} is the action selection function, f_{LO} is the function implementing the training observer to update the pair $[\varepsilon, eW]$, f_{LOT} is a function used by the learning observer to update table T, N is the number of iterations, bT is the best policy. The state of the system is estimated from the power error, wind speed and their derivatives. The reward function is defined as a sign function, the sign of which depends on the power error derivative. A learning observer is a control strategy module that receives an output power error and an active policy as input. It generates the best policy and adjusts the input parameters. The action selection module selects a random action with probability ε and a greedy action with probability $1 - \varepsilon$ to eliminate local minima.

NN is suitable for a broad class of problems in which there may be complete uncertainty of the dynamics of the agent and the environment. An example of the control of such a complex high-order nonlinear system described by partial differential equations is considered in [45]. The authors present the structure of predictive control based on a deep learning model. Gunther et al. [46] show how to perform a laser welding process while simultaneously using NN to predict the quality of a weld from processed video images and the laser power that depends on it. The actor-critic NN follows a stochastic control policy $\pi(a|s)$ chosen from a Gaussian $N(\mu, \sigma^2)$ distribution, where μ and σ are linear combinations of weight vector and feature vector. The assessment of the weld integrity was chosen as the measure of remuneration:

$$r = -0.5 + 1/[1 + \exp(|d^* - d|], \tag{22}$$

where d and d^* are the achieved and specified welding depth, respectively. The sigmoid form of the (22) function is limited to the range $[-0.5, 0]$ to ensure the convergence of the algorithm. At the same time, the critic evaluates the performance of the algorithm, after which the value function and weights are updated.

Lin and Yang [47] presented a control system for an autonomous wheeled mobile robot, which consists of an adaptive neural network speed control loop and a self-learning trajectory tracking loop, taking into account dynamic constraints. A three-layer NN consisting of an actor, a critic, and a shadow critic, in which each hidden neuron has a hyperbolic tangential activation function [48]:

$$\overline{y}_j(n) = a \tanh(b \sum_{i=0}^{I} w_{ji}(n)\overline{x}_i(n)), \overline{x}_0(n) = 1, (a, b) > 0, \tag{23}$$

Each output neuron has a linear activation function:

$$\overline{z}_k(n) = c \sum_{j=0}^{J} w_{kj}(n)\overline{y}_j(n), \overline{y}_0(n) = 1, c > 0 \tag{24}$$

In addition to activation functions, NN contains such components of their partial derivatives as two sensitivity functions ($\partial\overline{z}_k(n)/\partial w_{kj}(n)$, $\partial\overline{z}_k(n)/\partial w_{ji}(n)$) and the Jacobian function ($\partial\overline{z}_k(n)/\partial x_i(n)$). The task of the NN is to predict the state of the control object model and calculate the value of the Jacobians. The following utility function is used to keep track of the target speed:

$$U(n) = 0.25(v(n) - v_d(n))^2 + 0.25(\omega(n) - \omega_d(n))^2, \tag{25}$$

where v_d, ω_d are given linear and angular speeds, respectively. To achieve the control goal, it is necessary to find the NN weights corresponding to the minimum current and future values (25), for which it is required to minimize the functional of the form:

$$J(n) = \sum_{k=0}^{\infty} \eta^k U(n+k) = U(n) + \eta J(n+1), \tag{26}$$

where η is the discount factor, $0 < \eta \leqslant 1$. The NN is trained using the gradient descent method:

$$\Delta\omega_{km}(n) = \alpha\frac{\partial J(n)}{\partial w_{km}(n)},$$
$$\Delta w_{sm}(n) = \beta\frac{\partial E(n)}{\partial w_{sm}(n)},$$

where α, β is the learning rate, w_{km} is the weight of the m-th neuron associated with the k-th actor output, E is the shadow critic's NN correction error, w_{sm} is the weight of the m-th neuron associated with the s-th output of the shadow critic. The self-learning mechanism is built by identifying a special inverse velocity and angular model of the robot. After calculating the inversion error, the weights in the inverse velocity model are adjusted. Thus, the robot received control without a controller, training samples, and logical control rules.

Many studies using NN are devoted to the automation of manned and unmanned aircraft systems through the implementation of adaptive flight control strategies [49,50]. This is especially true for autonomous operation in difficult atmospheric conditions and system failures. Machine learning methods improve control accuracy and respond to unknown non-linear flight conditions. As shown in [50] NN, in this case, can be used to implement a trainable flight control system. Large groups of neural controls are trained to the desired behavior based on the measured data. The quality of training can then be assessed by statistical methods.

Due to their learning abilities, NNs are well suited for solving complex problems of nonlinear control and fault tolerance. Kruger et al. [51] proposed a non-linear adaptive flight control approach with feedback linearization and online learning in a sliding mode. The sliding-mode learning algorithm is based on the variable structure control theory and considers the NN and its learning as a control process. This approach allows the aircraft to overcome the negative impact of nonlinearities in the dynamics and external disturbances. In [52], a programmable logic integrated circuit of the field-programmable gate array (FPGA) type in a sliding mode based on a recurrent wavelet-neural network for controlling a linear ultrasonic motor is considered. To estimate nonlinear functions in the system model, system parameters, and external disturbances, the NN is used. The algorithm for adaptive learning of the NN in real-time is obtained using the Lyapunov stability theorem. The high efficiency of the algorithm has been verified for positioning and tracking problems. An adaptive sliding mode based on the support vector machine with fuzzy rules for improving the speed and accuracy of controlling a two-link robotic arm is described in [53]. The capabilities of this method allow good processing of non-linear signals and have the ability to generalize, which is used to reduce chatter during sliding control. The controller parameters are optimized using a hybrid learning algorithm that combines the least-squares method and the genetic algorithm.

Wang et al. [54] described an approach to designing a PID controller for a system with delay using a new NN-based Smith predictor, which makes the controller adaptive to changing system parameters. In [55,56], spike NNs are used to develop a controller for a highly nonlinear control system, which uses time-based pulse trains and can perform faster and more complex calculations. Spike NN controllers are capable of online learning and self-adaptation when moving from simulation to the real world. Such NN structures have fewer neurons and neural connections compared to previous generations of NN. The training of the spike NN in the framework of this work was carried out using evolutionary algorithms. The efficacy of the proposed approach was tested on a model of the movement of a robotic arm.

3.2. Controller Tuning

The use of PID, adaptive and fuzzy controllers is typical in the industry. Adaptive control methods make it possible to change both the structure and parameters of the controller depending on changes in the parameters of the control object or external disturbances. A historical perspective on adaptive control and learning methods is presented in the review [8]. The controller structure is often fixed, and only their parameters need to be adjusted. Methods for tuning the controller using a description of the system dynamics are known. Thus, they are demanding to obtain in reality since deep knowledge of the system or extensive measurements in a potentially open loop is required. The first works in this area describe an algorithm for tuning the PID controller through the transient response of the plant model and by the amplitude and period of self-oscillations of a closed-loop system [57,58]. Later, adaptive PID controllers with an identifying adaptive control algorithm were proposed, the coefficients of which are tuned based on model parameter estimates [59,60]. Sometimes only feedback measurements are possible, especially when the system is unstable. In this connection, their adjustment becomes laborious and inefficient in changing the system operating conditions. Therefore, they resort to automated methods that allow you to quickly determine the controller parameters by the task without human intervention. Work in this area began with self-adjusting structures. Work in this area began with self-adjusting structures. Dong and Tseng [61] proposed a multi-parameter self-adjusting controller for controlling the technological process of injection moulding machines. The system dynamics are described by the following stochastic discrete-time model:

$$A(q^{-1})y(t) = B(q^{-1})u(t-d-1) + C(q^{-1})e(t), \tag{27}$$

where $y(t)$ is an output vector of dimension p, $u(t)$ is an input vector of dimension s, $e(t)$ is white Gaussian noise of dimension p, q^{-1} is the reverse shift operator, $(q^{-1})y(t) = y(t-1)$, d is unit time delay. Using the self-tuning forecast procedure with k-step recursion, model (27) reads as:

$$\hat{y}(t+k|t) = \sum_{i=1}^{n_a} \hat{A}_i \hat{y}(t+k-i|t) + \sum_{i=d}^{n_b} \hat{B}_i u(t+k-i) + \sum_{i=k}^{n_c} \hat{C}_i \hat{e}(t+k-i), \tag{28}$$

where $k = 1, 2, \ldots, d$, $\hat{A}_i$, $\hat{B}_i$, $\hat{C}_i$ are matrix estimates for system (27). Then the optimization problem is written as:

$$J = \left\| D_0 \hat{B}_d u(t) + \hat{L}(t) \right\|_{Q_1}^2 + \left\| G_0 u(t) + \sum_{i=1}^{r} G_i u(t-1) \right\|_{Q_2}^2. \tag{29}$$

Therefore, the problem of stochastic optimization turns into the deterministic one: $\partial J / \partial u(t) = 0$, where $u(t)$ is the output of the self-tuning controller:

$$u(t) = -[(D_0 \hat{B}_d)^{\mathrm{T}} Q_1 D_0 \hat{B}_d + G_0^{\mathrm{T}} Q_2 G_0]^{-1} [(d_0 \hat{B}_d)^{\mathrm{T}} Q_1 \hat{L}(t) + G_0^{\mathrm{T}} Q_2 \sum_{i=1}^{r} G_i u(t-1)]. \tag{30}$$

Simulation results also indicate that additional control and coordination functions of the learning controller structure should be used to meet the requirements of more complex machines.

In work [62], devoted to self-adjusting controllers, an algorithm is analyzed and obtained by combining the least-squares estimates (LSE) with the adjustment of the minimum oscillations, obtained by means of the dynamics model. The main result is two theorems assuming that the parameter estimates converge and defines the closed-loop system. The first theorem states that some output and intersecting covariance of output control variables will disappear under small assumptions in the registries. In the second theorem, it is accepted that the controlled system is a general linear stochastic system of the nth order. It is shown that if the estimates of the parameters converge, then the obtained regulation law is the control law of the minimum oscillation, which could be calculated with the known parameters.

Despite its simplicity, PID controllers do not lose their efficiency and continue to be used in modern control systems, and their parameters are now tuned using machine learning methods. In [63], the heuristic harmony search method is applied to PID controller tuning for a static reactive power compensator control system connected to a wind energy conversion system.

Schillinger et al. [64] discussed the automated tuning of the proportional-integral (PI) controller tuning for a fuel supply system in a car. The paper compares two tuning methods: secure active learning and Bayesian optimization. Safety implies restrictions on such parameters as, for example, overshoot, which can lead to loss of system stability and emergency operation. Active learning involves minimizing the loss function, which depends only on the controller parameters, limits a high control error and oscillatory control actions:

$$J = \max(J_1, J_2), \quad J \in [0, 1], \tag{31}$$

$$J_1 = \frac{1}{n}\sum_{k=1}^{n}(1 - \exp(-\frac{\theta_1}{2}(y(k) - r(k))^2)), \tag{32}$$

$$J_2 = \frac{1}{n}\sum_{k=1}^{n}(1 - \exp(-\frac{\theta_2}{2}(u(k) - \overline{u}(k))^2)), \tag{33}$$

where $r(k)$, $u(k)$, $y(k)$ are the discrete-time input signal, controller and controlled plant outputs, respectively; $\theta_{1,2}$ are the adjustable parameters; $\overline{u}$ denotes the moving average of the controller output. To train the discriminant model, the following risk function h is defined:

$$h = \min(h_1, h_2), \tag{34}$$

$$h_1 = \frac{\max_{k\in[1,n]}(y(k)) - y_{max}}{y_{\min\max} - y_{max}}, \quad h_2 = 1 - \frac{J}{J_{\max}}, \tag{35}$$

where $y_{\max}$ is the maximum allowed output value, $y_{\min\max}$ is the smallest expected maximum output value, $J_{\max}$ is the maximum allowed loss function value. The goal of the Bayesian optimization problem is to find the global minimum or maximum of the functional at a high cost for its evaluation. In [64], the Lower Confidence Bound (LCB) is chosen for minimization:

$$LCB(x) = \mu_{f*}(x) - k\sigma_{f*}(x), \tag{36}$$

where μ_{f*} is the predicted value for f model, k is a tunable parameter, σ_{f*} is the predicted standard deviation for the regression. To ensure the selection of parameters that are safe for the system, Schillinger et al. [64] proposed a combined approach, which consists in using Bayesian optimization with a discriminant g model. The difference between this

approach and active learning lies in the restriction of the lower confidence limit (36) to the estimated risk:

$$x_{i+1} = \operatorname{argmin} LCB(x_*) \quad \text{when} \quad \mu_{g*}(x_*) - \nu\sigma_{g*}(x_*) \geqslant 0. \tag{37}$$

Both approaches have shown good results and can be further improved.

In [65], safe Bayesian optimization was also used to tune the PI controller of the quadrotor vehicle control system. The following algorithm for calculating the estimate of the functional with restrictions on the area of safe parameters is proposed:

$$J(a_n) = C(a_n) - 0.95C(a_0), \tag{38}$$

$$C(a_n) = -\sum_{k=0}^{N} \left(x_k^{\mathrm{T}} \mathrm{Q} x_k + \mathrm{R} u_k^2\right), \tag{39}$$

where a_n are controller parameters; x denotes the state vector; Q, R are positive semi-definite matrices; N is the prediction time horizon; u is the system input; n is the iteration number .

In [66] the parameters of the internal combustion engine were tuned based on the MDP in real-time to achieve the optimal operating mode for any driving style.

In [67], instead of a metaheuristic algorithm, a more efficient support vector machine is proposed for training and building a model that predicts the optimal parameters of a resonant vibration damper. However, the adequacy of these models is highly dependent on the number and accuracy of the databases used for training. Good results were shown by the genetic algorithm when tuning the autopilot PID controller [68]. To model and predict the dynamic characteristics of a relay-reactive machine, [69] proposed to use an adaptive fuzzy neural system based on radial basis functions. In the considered system, only the phase flux linkage is available for indirect measurement, by which the phase torque is calculated. The learning of the machine to know the electromagnetic characteristics is carried out using a hierarchical self-organizing algorithm.

Another direction of artificial intelligence application for PID controller design is presented in [70–73]. Wen et al. [73] proposed a trajectory control structure using reinforcement learning of a fuzzy controller. The robot dynamics and the controller structure are described by the following IF-THEN fuzzy rules:

$$\text{if} \quad x_1(t) = M_1^i \text{ and } x_2(t) = M_2^i \text{ and } x_n = M_n^i, \tag{40}$$

$$\text{then} \quad \dot{x}(t) = A_i x(t) + B_i u(t), \tag{41}$$

$$u(t) = K_j x(t), \quad i = 1, 2, ..., r \; j = 1, 2, ..., r, \tag{42}$$

where $x(t)$ is the state vector for the i-th rule, M_j^i is the fuzzy set, $u(t)$ is the input vector, r is the number of fuzzy rules, A_i, B_i are matrices, K_j denotes the feedback coefficient for j-th linear subsystem. To train the controller, an iterative Q-learning algorithm is used, due to the presence of an agent that selects the optimal sequence of MDP actions. The calculation algorithm evaluates each value of the Q-function for each state-action pair (s_t, a_t). Q-function is updated according to the following law:

$$Q(s_t, a_t) = r_t + \gamma Q(s_{t+1}, a), \tag{43}$$

where r_t is the reward for the performed action, γ is the discount factor. Depending on the obstacle area S_d, the desired position S_t and the safe area S_s, the reward function has the form:

$$r = \begin{cases} 100, & S = S_t, \\ 0 & S \in S_s, \; S \neq S_t, \\ -100, & S \in S_d. \end{cases} \tag{44}$$

The PID controller built based on the fuzzy logic method with a Gaussian membership function showed its efficacy in controlling a stepper motor [70]. The fuzzy structure of the controller allows its gain to be adapted to changes in the system. An analogous approach was used for an active magnetic bearing in [72] where the centres of triangular membership functions and the gain were optimized additionally using a genetic algorithm. Mitra et al. [71] used a zero-order Sugeno fuzzy model, in which parameters are tuned offline using a hybrid learning algorithm. This approach is a combination of least-squares and backpropagation. A computer simulation of a voltage stabilizer system showed that the fuzzy model satisfactory ensures the system stability at small signals.

3.3. Identification Problems

Ercin and Coban [74] presented an approach to identifying a linear system described by difference equations using the bee swarm algorithm. The identification problem is formulated as an optimization problem, the purpose of which is to obtain a model and a set of parameters for which the prediction error between the actual object and model outputs will be minimal. The results of the algorithm operation are demonstrated on a direct current (DC) motor model. Atashpaz Gargari et al. [75] used a heuristic colonial competitive algorithm to tune the PID-controller parameters of the evaporator control system while minimizing the absolute quadratic system tracking error. This algorithm's efficacy is demonstrated related to the Ziegler-Nichols method and the genetic algorithm. Rahimi et al. [76] used a heuristic competitive algorithm to identify the parameters of a permanent magnet synchronous motor model in real-time. To this end, the mean square error of monitoring the system state vector is minimized.

The trend in intelligent control design is aimed at a combination of various methods. Thus, in [77], the Kalman filter, recurrent NN, and gradient descent method are used to identify the robot parameters. The obtained control system stability is estimated by the Lyapunov function. The robot dynamics are described by the Euler–Lagrange equations. The Cartesian forces are taken as the system outputs, position X, velocity $\dot{X}$ and the acceleration $\ddot{X}$ constitute the input regressor vector. For the problem of reverse dynamic identification, three self-learning NNs of direct action with a multilevel structure of neurons are considered. The NNs include the following components: one hidden layer, a sufficient number of non-linear neurons and an output layer with a linear activation function. Quality function J to control the rules for updating the NN weights is taken as the mean square value of four errors:

$$J = e_1^{\mathrm{T}} e_1 + e_5^{\mathrm{T}} e_5, \text{ if } e_2 = e_3 = e_4 = 0, \tag{45}$$

$$\begin{cases} e_{1i} = \tau_i - \hat{M}_i \ddot{X} - \hat{C}_i \dot{X} - \hat{G}_i, \ i \in \{1, 2, ..., n\}, \\ e_{2i} = \dot{\hat{M}}_{ii} - 2\hat{C}_{ii}, \ i \in \{1, 2, ..., n\}, \\ e_{3m} = \dot{\hat{M}}_{ji} - 2\dot{\hat{M}}_{ij} - 2(\hat{C}_{ji} + \hat{C}_{ij}), \ i \neq j, \ m \in \{1, 2, ..., (n^2 - n)/2\}, \\ e_{4i} = \mid \hat{M} - \lambda I_{n \times n} \mid, \ \lambda = \mathrm{eig}(\hat{M}) \exp(\lambda_0 / t), \ i \in \{1, 2, ..., n\}, \\ e_5 = X(k) - (T_s \hat{\dot{X}}(k) + \hat{X}(k-1)), \end{cases} \tag{46}$$

$$\begin{cases} \hat{\dot{X}}(k) = T_s \hat{\ddot{X}}(k) + \hat{\dot{X}}(k-1), \\ \hat{\ddot{X}}(k) = \hat{M}^{-1}(k-1)(\hat{\tau}(k) - \hat{C}(k-1) - \hat{G}(k-1)), \end{cases} \tag{47}$$

where $\hat{M}$, $\hat{C}$, $\hat{G}$ are the estimates of the inertia matrices components, Coriolis, centrifugal, and gravitational components, respectively; e_5 is the prediction error, $\hat{\tau}$ is the estimate of the applied forces vector; T_s denotes the sampling rate; $\lambda > 0$ is the parameter for ensuring the convergence time of the algorithm; λ is the vector of the system eigenvalues, $I_{n \times n}$ is the identity matrix. NN for M, C, G are configured according to the following principle:

$$\hat{M}_{ij} = W_M^2 f(W_M^1 u), \ u = [X, I_{n \times n}]^{\mathrm{T}}, \tag{48}$$

$$\hat{M}_i = [M_{i1}, \ldots, M_{ij}, \ldots, M_{in}]^{\mathrm{T}}, \ i, j = \{1, 2, \ldots, n\}, \tag{49}$$

where W_M^1, W_M^2 are the weights of the hidden and output NN layers. Agand et al. [77] considered two learning algorithms: based on the error backpropagation and the discrete Lyapunov function. It is proved that the following Lyapunov function guarantees the global stability of the system:

$$V(k) = \sum_i \sum_j || W_j^i ||_F^2, \quad i \in \{o, h\}, \; j \in \{M, C, G\}. \tag{50}$$

3.4. Optimization Problems

A common optimization problem is to find a certain extreme value of a system parameter. So, for example, in [78] the purpose of the controller synthesis is stabilizing the system energy, which is expressed in providing maximum damping by the following functional: $J = \max(\min(\xi_{ij}))$, where i are the eigenvalues of the closed-loop system, j are the working points. The following standard lead-lag controller is used to stabilize the system energy

$$K(s) = K_p \frac{sT_w}{1 + sT_w} \cdot \frac{1 + sT_1}{1 + sT_2} \cdot \frac{1 + sT_3}{1 + sT_4}, \tag{51}$$

where T_1, T_2, T_3, T_4 are time constants, K_p is the gain, $s \in \mathbb{C}$ stands for the Laplace transform variable. The search for these parameters was made by the method of population incremental learning, which combines the algorithm of optimization and distribution estimation.

In [79], the dynamic model of a soft robot is described using a recurrent NN that recursively defines the robot trajectory as follows:

$$x_{i+1} = f(x_i, x_{i-1}, \tau_i), \quad i = 1...k, k = t_f/dt, \tag{52}$$

where x_i, x_{i-1}, x_{i+1} are the current, previous and next position of the manipulator, τ_i are the control forces of the electric cables, f is the trained comparison of the NN , dt denotes the time step. The optimal strategy of manipulator actions is obtained by solving the following non-linear optimization problem:

$$\Pi(t)^* = \min_{\tau} \parallel x^* - x \parallel^2 + \sum_k \tau_k^{\mathrm{T}} R \tau_k, \quad \text{as } 0 \leqslant \tau_k^m \leqslant \tau_{max}^m, \tag{53}$$

where m is the number of manipulators, x^*, and x is the desired and actual position coordinate of the robot. Problem (53) is solved numerically using the gradient method of sequential quadratic programming (SQP) [80].

Gradient-free search algorithms are extensively used because of their universality for all optimization problems. It is also applied for NN since they do not use the function gradient and do not assume its differentiability [81]. Their peculiarity is that the ensuing solution to the optimization problem is acceptable but not optimal. Recently, various bio-inspired solutions have become popular, the ideas of which are borrowed from nature, cf. [82,83]. These include evolutionary, swarm and colony algorithms [84,85], population [86] etc. Also, the *bat algorithm* [87] is known, related to swarm intelligence based on echolocation. The *cuckoo swarm* algorithm is used for the PID controller tuning in the DC motor control system [88] and thyristor-controlled longitudinal compensation [89]. After comparing the swarm algorithm with the heuristic one, the former proved to be more efficient.

Wang et al. [90] proposed an optimal control approach based on a support vector algorithm to minimize the power consumption of walking bipedal robots under an unknown system dynamic model conditions and small data sample sizes. The new controller is built into the optimal controller and aims to minimize the energy-related cost function with constraints on robot articulation angles. The energy functional is as follows:

$$J_{EE} = \int_0^T \frac{1}{2} \tau^{\mathrm{T}} \tau dt, \quad \text{at} \quad \tau = g(\Theta), \tag{54}$$

where Θ is a vector of generalized coordinates, $g(\cdot)$ is parametrized by NN. The quality function in the support vector machine in quadratic form is:

$$J_{SVM} = \min \frac{1}{2} w^{\mathrm{T}} w + \frac{1}{2} C \sum_{i=1}^{N} \xi_i^2, \text{ as } \tau_i = w^{\mathrm{T}} \varphi(\Theta_i) + \xi_i, \tag{55}$$

where w is a vector of weights, ξ_i is a positive variable, N is the number of training samples, C is a penalty factor, $\varphi(\cdot)$ is a mapping function of the input space into that of higher-order features. The resulting functional consists of the sum $J_{EE} + J_{SVM}$.

In [91], multi-criteria optimization of the PID controller parameters was carried out using an improved "teaching–learning" search algorithm that excludes the function values from falling into a local minimum. For this, the learning phase is no longer limited to two learners and is combined with an additional state. Also, it is combined with a blocked device phase, in which parameters close to incompatible targets are blocked. This results in each goal not being able to conflict with the others. In [92], a benchmarking study result is represented for the PID controller parameters optimization of the DC motor control system using the bee colony algorithm, the particle swarm method and "teaching–learning". The latter outcome has shown the best results.

3.5. Problems of Iterative Learning

One form of artificial intelligence is machine learning, which allows a system to learn from data stored in memory instead of being explicitly programmed [93]. More accurate models are created based on the training data sets processing. It allows training models both before they are used and continuously. The iterative process of training models always grants for improving the types of relationships between data elements, regardless of their complexity or high volume. The off-line trained model can be used to continue training in real-time.

Repetitive control was originally formulated in [94] for overcoming repetitive disturbances problems within a known period and characteristic of the driver rotation. Then, it was further developed for cases where the period of disturbances is unknown or varies in time. The repetitive control problem is defined as follows. An asymptotically stable discrete-time system is considered:

$$A(z^{-1}) y(k) = z^{-d} B(z^{-1}) [u(k) + w(k)], \tag{56}$$

where u, y, w are input, output and perturbation respectively, z^{-1} is one time step delay, $A(z^{-1}) = 1 + a_1 z^{-1} + ... + a_n z^{-n}$, $B(z^{-1}) = b_0 + b_1 z^{-1} + ... + b_m z^{-m}$. Perturbation w repeats with period N:

$$(1 - z^{-N}) w(k) = 0. \tag{57}$$

The goal of the control law is the asymptotic approach of the control error to zero: $\lim_{k \to \infty} e(k) = 0$. The regulator can be synthesized in various ways, the most popular and studied of which is the method based on the principle of the internal model. It consists in constructing independent blocks for identifying harmonic parameters [95].

Tomizuka [96] presented some of the fundamental problems of designing repetitive control controllers for handling periodic disturbances arising in electromechanical systems. Overcoming the parametric uncertainty of perturbations is proposed with the help of adaptive control tools, μ-synthesis, Q-parametrization, and anti-windup compensation.

Since, in the repetitive control, the repetitive frequency components from the fundamental to higher harmonics are compensated, with a large number of components ($N = 1000$), the problem of computational costs arises.

Closely related to iterative control is iterative learning control [97–99]. Iterative learning control is an intelligent tracking control technique for repetitive systems to improve the system's accuracy. For it, two processes are distinguished: the repeated execution

of homogeneous actions and their sequential adjustment. Examples of such systems are robots in industrial conveyor systems, nanosystems, chemical batch processes, reliability test facilities, etc. It was designed to manage repetitive operations over a finite amount of time that robots must perform. Generally, iterative learning can be described as a system of linear state equations, one of which represents the dynamics of the system in time, the second—the dynamics of iterative learning [100,101]:

$$x(t+1,k) = Ax(t,k) + Bu(t,k), \quad y(t,k) = Cx(t,k), \tag{58}$$

where t is discrete time, $t = 0, 1, \ldots T$; k denotes the learning step number, $k = 0, 1, \ldots$; $u(t,0) = 0$, $x \in R^n$ is the state vector, $x(0,k) = x_0$; $y \in R^p$ denotes the output vector, $u \in R^m$ is the control vector; A, B, and C are matrices of appropriate dimensions.

The iterative learning algorithm is based on the fact that the measured system variables are written to the program memory at each iteration. At each step, these variables are processed to achieve the specified quality of the control criterion. The control goal is to achieve the specified value y_r by the vector of output parameters y with the specified accuracy. In connection with this, the law of iterative learning control with correction at each step is written as:

$$u(t,k) = u(t,k-1) + \Delta u(t,k-1), \tag{59}$$

where Δu is a corrective addition to the input variable of the system, chosen from the condition that the error

$$e(t,k) = y_r(t) - y(t,k), \tag{60}$$

tends to zero as $k \to \infty$, while $u_\infty = \lim\limits_{k\to\infty} u_k(t)$ is bounded in absolute value.

Seel et al. [102,103] proposed an approach to the iterative learning control of a neuroprosthesis, which forms the law of changes in the intensity of electrical stimulation at each moment of movement. Foot angle α is estimated in real-time using gyroscopes and accelerometers. At the beginning of each next human step, the mismatch of the foot inclination angle e_j is determined as an element-by-element deviation between the measured angular trajectory a_j and the first n_j readings of the corresponding reference trajectory r_α: $e_j = r_\alpha(1 : n_j) - \alpha_j$.

Mohammadpour et al. [104] presented a fault-tolerant control method based on iterative current loop learning control to restore the performance of multi-phase permanent magnet drives under open-circuit failure conditions. The advantage of this approach is that no-fault finding and diagnosis are required, but only torque measurement is sufficient. Thus, iterative learning management provides comprehensive information about the system and provides stability to model uncertainties. For the movement of a system of cloud wheeled robots along a given trajectory and simultaneous transportation of cargo, a flexible control scheme for trajectory tracking using iterative learning control has been developed, and an analysis of the system stability [105] has been carried out. Warrier and Devasia [106] considered human-led iterative learning for trajectory tracking tasks when the controller receives data from the actions of a human operator. Hladowski et al. [99] presented new results on the dynamic development of iterative learning control laws considering the influence of noise.

In [5], an iterative procedure for planning the milling process is represented, for which it is necessary to know the technological parameters of the machine and the geometric parameters of parts to form the trajectory of the machine tool. The tool deviation is a significant problem that requires constant checking and planning of the milling process. Dittrich et al. [5] proposed the following solution, which reduces the processing error to 50%: using machine learning methods, they predict the error between the model of the processed form and measurements of the actual surface. For this, a statistical support vector machine is used, and the data set from previous processes is used as a training data set.

Silvestri et al. [107] presented the results of a study on self-optimizing measuring tools to develop and validate innovative manufacturing equipment and control systems based on understanding, evaluating and controlling the production characteristics of large machine tools with a self-learning controller capable of managing a large amount of measured data from sensors. The proposed architecture uses the support vector machine and fuzzy logic to compare the collected data and the actual current state of the process.

3.6. Problems of Forecasting

A separate class of problems includes the problems of predicting the state of the system, which requires the extensive collection and subsequent processing of information from measuring sensors. Kanawaday and Sane Kanawaday and Sane [108] proposed to apply the Internet of things (IoT) technology in production, which allows data analytics to obtain fruitful information. Prediction of potential slitting machine failures is done using a time series analysis methodology called moving average autoregression. Often, systems include preventive measures, such as the emergency shutdown of the load when there are surges in the operating frequency in the power system. In [109], this problem is solved by training the prediction model against a pre-prepared database and then applying it in real-time. An actual problem for processing machines is tool wear. In [110], is proposed an approach to creating a model of the grinding process for controlling material removal using a robotic belt grinder that can track changes in operating conditions and accurately predict material removal. The main idea is to build an initial support vector regression model based on the grinding data as training samples. The trained model is modified according to the real-time measurement data, and the machine control parameters are calculated. Geometric shape prediction has proven to be a necessary component of soft robot control. Intelligent modelling of the NN-based turning process for predicting machining quality characteristics for known combinations of cutting parameters is described in [111]. The best results were shown by recurrent NN and in the bee colony. In [112], based on measurements of piezoresistive sensors, a basis for predicting the three-dimensional configuration of a soft robot using deep learning is proposed. Sharaf et al. [113] showed the use of machine learning methods to assess the condition of the downhole pump and its subsequent diagnostics. The data were collected using a dynamometer measuring the load on the sucker rod and processed using a specialized software package for developing and ensuring the real-time operation of systems for collecting and processing information about the monitoring object. Thus, the forecast implementation based on machine learning is a modern and effective way to prevent accidents in production, reduce maintenance costs and improve the overall production process.

In addition to classifying papers according to the machine learning algorithms, several surveys are devoted to learning applications for a specific technology. The review paper [114] discusses some machine learning methods that can be used to develop orbital and spatial control of spacecraft. The authors rightly noted that there are very few works on finding the optimal control trajectory compared to the works devoted to NN. The publications are divided according to the method used: supervised and reinforcement learning. Also, among the review publications, one can single out works in which machine learning methods are applied to a specific area of technology: control of bipedal robots [115], electric power systems [116], rotating electric machines [117], a vibration of structures [118], autonomous vehicle control [119], in various areas of mechanics [120], vehicle suspension control [121].

4. Conclusions

Existing in the world trend of transition to high-tech types of production organization is reflected in scientific publications on the application of intelligent methods in control of electro-mechanical systems. Table 1 shows the intellectual methods considered in this paper, application area, advantages and disadvantages.

Table 1. Summary table of learning methods.

Learning Methods	Application and References	Advantages	Drawbacks
Supervised learning			
Support vector machine	Stabilization and program control [53]; Controller tuning [66]; Optimization [90]; Iterative learning [5,107]; Forecasting [110];	More efficient in multidimensional spaces, the number of dimensions should be greater than the number of samples, does not require a lot of memory.	Clear labeling of data is required, no probabilistic explanation for the classification, not suitable for large and noisy datasets.
Human-guided	Iterative learning [106]	A human can reproduce a unique complex experience.	A human can be wrong.
Gradient boosting machines	Forecasting [113]	High prediction accuracy, suitable for optimizing many loss functions and hyperparameters, no data preprocessing required, imputation not required.	Can lead to overemphasize outliers and cause overfitting, compute costs.
Classical reinforcement learning			
Policy gradient	Stabilization and program control [37]	Stochastic policy training, not required Q-function (V-function only), more effective in high dimensional action spaces.	Only on-policy, converge on a local maximum rather than on the global optimum, high variance, may be unrealistic reward, are sensitive to network initialization.
Actor-critic	Stabilization and program control [46,47]	Convergence is guaranteed, reduce variance with respect to pure policy search methods, not required Q-function (V-function only).	Only on-policy, the trend converges to local extremes.
Soft actor-critic	Stabilization and program control [38]	Bypassing local extrema.	Only on-policy, only for the task of continuous control, it is difficult to choose the temperature parameter.
Bayesian	Stabilization and program control [38]; Controller tuning [63,64]	The probability of transitions is known, the possibility to include preliminary knowledge in the algorithm.	Sampling from a Bayesian distribution using heuristics only.
Evolutionary search algorithms	Controller tuning [53,62,67,68,73]; Identification problems [74,76]; Optimization [88,89,91,92]; Forecasting [111]	Suitable for a wide range of applications, suitable for any cost function, combinable with other methods.	The lack of a guarantee of finding the optimal solution in a finite time, heuristic, high resource intensity, require adaptation to a specific class of problems.
Markov decision problem	Stabilization and program control [41]; Controller tuning [65]	No assumptions about the form of the optimal policy, can handle infinite and finite horizons are needed.	Transition probability unknown, policy space may be too large (“curse of dimensionality”), learns from the completely observable states only, only policy iteration.

Table 1. *Cont.*

Learning Methods	Application and References	Advantages	Drawbacks
	Neuron networks		
Recurrent	Identification problems [75]; Optimization [77]; Forecasting [111]	Simple structure, can process inputs of any length, has internal memory.	Gradient exploding and vanishing problems, needs to training.
Deep recurrent	Stabilization and program control [45,52]; Forecasting [112]	Work with complex raw data, have long short term memory.	Time for training, cannot be stacked up ,the vanishing or exploding gradient problem, difficult to process longer sequences.
Pulsed	Stabilization and program control [55,56];	High speed of transmit data, can compute any function.	Hard to train, only a large neural network is effective.
Extreme learning machine	Forecasting [109];	Short training time.	Cannot be “deep", low evaluation speed.
	Adaptive methods		
	Controller tuning [59–61]	Online, works for unknown systems.	Not optimal, there are applicability limits.
	Fuzzy logic methods		
Takagi–Sugeno	Controller tuning [69–73]; Iterative learning [107]	Robust to inputs, required low memory space, flexible, simple structure.	Low accuracy, no systematic approach, dependent on human knowledge and experience, updating of the rule is required, many tests are required to verify their work.

AI methods make it possible to achieve the goal and solve the tasks of controlling mechatronic systems that previously seemed impossible, while with greater computational efficiency and ease of implementation. The complexity of control tasks for essentially nonlinear, uncertain, or multi-agent systems under the influence of the external environment requires an individual approach to solving a specific problem, for which a large number of tools have already been proposed. The effectiveness of these algorithms for teaching complex systems is established, as a rule, only during a practical experiment. The development of the algorithms themselves is aimed not only at increasing the speed and accuracy of learning, but also at increasing independence from a learning strategy rigidly set by a person and its adaptation to various goals. Developers borrow ideas from nature, trying to reproduce the behavior of living organisms in their algorithms. Future research sets itself the task, which is often called “learning-to-learn”, cf. [122] when the agent has to choose the learning strategy and adjust its meta-parameters.

Author Contributions: Conceptualization, B.A.; Data curation, I.Z.; Formal analysis, I.Z.; Funding acquisition, B.A.; Investigation, I.Z.; Methodology, B.A.; Project administration, B.A.; Software, I.Z.; Supervision, B.A.; Writing—original draft, I.Z. and B.A.; Writing—review & editing, I.Z. and B.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Science and Higher Education of the Russian Federation (Project No. 075-15-2021-573, performed in the IPME RAS).

Data Availability Statement: Not applicable

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

Abbreviations

The following abbreviations are used in this manuscript:

ASAC	Adaptive Soft Actor-Critic
ADPRL	Approximate Dynamic Programming Reinforcement Learning
AI	Artificial Intelligence
DC	Direct Current
FL	Fuzzy Logic
FPGA	field-programmable gate array
IoT	Internet of Things
LCB	Lower Confidence Bound
LSE	least-squares estimates
MPD	Markov Decision Process
NN	Neural Network
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
RL	Reinforcement Learning
SQP	Sequential Quadratic Programming
TDL	Temporal Difference Learning

References

1. Vepa, R. Review of techniques for machine learning of real-time control strategies. *Intell. Syst. Eng.* **1993**, *2*, 77–90. [CrossRef]
2. Mitchell, T. *Machine Learning WCB*; McGraw-Hill: Boston, MA, USA, 1997.
3. Mosheiov, G. Scheduling problems with a learning effect. *Eur. J. Oper. Res.* **2001**, *132*, 687–693. [CrossRef]
4. Gurel, S.; Akturk, M. Scheduling preventive maintenance on a single CNC machine. *Intern. J. Prod. Res.* **2008**, *46*, 6797–6821. [CrossRef]
5. Dittrich, M.A.; Uhlich, F.; Denkena, B. Self-optimizing tool path generation for 5-axis machining processes. *CIRP J. Manuf. Sci. Technol.* **2019**, *24*, 49–54. [CrossRef]
6. Matni, N.; Proutiere, A.; Rantzer, A.; Tu, S. From self-tuning regulators to reinforcement learning and back again. In Proceedings of the IEEE Conference on Decision and Control (CDC 2019), Nice, France, 1–13 December 2019; pp. 3724–3740. [CrossRef]
7. Fomin, V.N.; Fradkov, A.L.; Yakubovich, V.A. *Adaptive Control of Dynamical Plants [Adaptivnoye Upravleniye Dinamicheskimi Ob'Yektami]*; Fizmatlit: Moskva, Russia, 1981. (In Russian)
8. Annaswamy, A.M.; Fradkov, A.L. A historical perspective of adaptive control and learning. *Annu. Rev. Control.* **2021**, *52*, 18–41. [CrossRef]
9. Yakubovich, V. Some general theoretical principles of the construction of trainable identification systems. *Comput. Eng. Program. Issues* **1965**, 3–71. (In Russian)
10. Yakubovich, V. Recurrent finitely convergent algorithms for solving systems of inequalities. *Dokl. Akad. Nauk SSSR* **1966**, *166*, 1308–1312.
11. Bondarko, V.A.; Yakubovich, V.A. The method of recursive aim inequalities in adaptive control theory. *Intern. J. Adapt. Control. Signal Process.* **1992**, *6*, 141–160 [CrossRef]
12. Fradkov, A.L. Scientific School of Vladimir Yakubovich in the 20th century. *IFAC-PapersOnLine* **2017**, *50*, 5231–5237.
13. Gusev, S.V.; Bondarko, V.A. Notes on Yakubovich's method of recursive objective inequalities and its application in adaptive control and robotics. *IFAC-PapersOnLine* **2020**, *53*, 1379–1384. [CrossRef]
14. Lipkovich, M. Yakubovich's method of recursive objective inequalities in machine learning. In Proceedings of the 14th IFAC Intern. Workshop Adaptation and Learning in Control and Signal Processing (ALCOS 2022) IFAC, Casablanca, Morocco, 29 June–1 July 2022.
15. Perel'man, I. Analysis of modern adaptive control methods from the stand-point of application to automatization of technological processes. *Avtom. Telemekhanika* **1991**, *7*, 3–32.
16. Andrievsky, B.; Fradkov, A. Speed Gradient Method and Its Applications. *Autom. Remote. Control.* **2021**, *82*, 1463–1518. [CrossRef]
17. Nikolenko, S.; Kadurin, A.; Arkhangelskaya, E. *Deep Learning. Dive into the World of Neural Networks [Glubokoye Obucheniye. Pogruzites' v Mir Neyronnykh Setey]*; Piter: Saint Petersburg, Russia, 2018; pp. 93–123. (In Russian)
18. Zadeh, L.A.; Klir, G.J.; Yuan, B. *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems*; World Scientific: Singapore, 1996.
19. Novak, V. *Mathematical Principles of Fuzzy Logic [Matematicheskiye Printsipy Nechetkoy Logiki]*; Fizmatlit: Moscow, Russia, 2006. (In Russian)
20. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.* **1989**, *2*, 303–314. [CrossRef]
21. Lee, J.; Shin, J.; Realff, M. Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Comput. Chem. Eng.* **2018**, *114*, 111–121. [CrossRef]
22. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2018.
23. Buoniu, L.; Babuska, R.; De Schutter, B.; Ernst, D. *Reinforcement Learning and Dynamic Programming Using Function Approximators*; Tailor & Francis Group: Abingdon, UK, 2010. [CrossRef]

24. Kurrek, P.; Jocas, M.; Zoghlami, F.; Stoelen, M.; Salehi, V. AI motion control—A generic approach to develop control policies for robotic manipulation tasks. In Proceedings of the International Conference on Engineering Design, ICED, Delft, The Netherland, 5–8 August 2019; pp. 3561–3570. [CrossRef]
25. Kim, J.B.; Lim, H.K.; Kim, C.M.; Kim, M.S.; Hong, Y.G.; Han, Y.H. Imitation reinforcement learning-based remote rotary inverted pendulum control in openflow network. *IEEE Access* **2019**, *7*, 36682–36690. [CrossRef]
26. Menghal, P.; Laxmi, A. Real time simulation: A novel approach in engineering education. In Proceedings of the ICECT 2011–2011 3rd International Conference on Electronics Computer Technology, Kanyakumari, India, 8–10 April 2011; Volume 1, pp. 215–219. [CrossRef]
27. Powell, W. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2nd ed.; John Wiley and Sons: Hoboken, NJ, USA, 2011.
28. Bertsekas, D.; Tsitsiklis, J. Neuro-dynamic programming: an overview. In Proceedings of the 34th IEEE Conference Decision and Control (CDC'95), New Orleans, LA, USA, 13–15 December 1995.
29. Prokhorov, D.; Wunsch, D. Adaptive critic designs. *IEEE Trans. Neural Netw.* **1997**, *8*, 997–1007. [CrossRef]
30. Werbos, P.J., Reinforcement Learning and Approximate Dynamic Programming (RLADP)—Foundations, Common Misconceptions, and the Challenges Ahead. In *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*; John Wiley and Sons, Ltd.: Hoboken, NJ, USA, 2012; Chapter 1, pp. 1–30. Available online: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118453988.ch1 (accessed on 7 July 2022). [CrossRef]
31. Lewis, F.; Vrabie, D. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits Syst. Mag.* **2009**, *9*. [CrossRef]
32. Puterman, M. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley and Sons: Hoboken, NJ, USA, 2014.
33. Sun, J.; Liu, F.; Si, J.; Guo, W.T.; Mei, S.W. An improved approximate dynamic programming and its application in SVC control. *Dianji Kongzhi Xuebao/Electric Mach. Control.* **2011**, *15*, 95–102.
34. Watkins, C. *Learning from Delayed Rewards*; University of Cambridge: Cambridge, UK, 1989.
35. Yoo, J. Gaussian process model based reinforcement learning. *J. Inst. Control. Robot. Syst.* **2019**, *25*, 746–751. [CrossRef]
36. Deisenroth, M.; Rasmussen, C.; Peters, J. Gaussian process dynamic programming. *Neurocomputing* **2009**, *72*, 1508–1524. [CrossRef]
37. Lopes, G.; Ferreira, M.; Da Silva Simoes, A.; Colombini, E. Intelligent control of a quadrotor with proximal policy optimization reinforcement learning. In Proceedings of the 15th Latin American Robotics Symposium, 6th Brazilian Robotics Symposium and 9th Workshop on Robotics in Education, LARS/SBR/WRE, Paraiba, Brazil, 6–10 November 2018; pp. 509–514. [CrossRef]
38. Kim, J.; Hong, M.; Lee, K.; Kim, D.; Park, Y.L.; Oh, S. Learning to Walk a Tripod Mobile Robot Using Nonlinear Soft Vibration Actuators with Entropy Adaptive Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2020**, *5*, 2317–2324. [CrossRef]
39. Puccetti, L.; Kopf, F.; Rathgeber, C.; Hohmann, S. Speed Tracking Control using Online Reinforcement Learning in a Real Car. In Proceedings of the 6th International Conference on Control, Automation and Robotics (ICCAR), Singapore, 20–23 April 2020; pp. 392–399. [CrossRef]
40. Poon, C.S. Sensorimotor learning and information processing by Bayesian internal models. In Proceedings of the 26th Annual Intern. Conference of the IEEE Engineering in Medicine and Biology (EMBC), San Francisco, CA, USA, 1–5 September 2004; Volume 2, pp. 4481–4482. [CrossRef]
41. Endo, Y. Anticipatory robot control for a partially observable environment using episodic memories. In Proceedings of the IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 2852–2859. [CrossRef]
42. Sun, L.; You, F. Machine Learning and Data-Driven Techniques for the Control of Smart Power Generation Systems: An Uncertainty Handling Perspective. *Engineering* **2021**, *7*, 1239–1247. [CrossRef]
43. Tomin, N.; Kurbatsky, V.; Guliyev, H. Intelligent control of a wind turbine based on reinforcement learning. In Proceedings of the 16th Conference on Electrical Machines, Drives and Power Systems (ELMA 2019), Varna, Bulgaria, 6–8 June 2019, pp. 1–6. [CrossRef]
44. Sierra-Garcia, J.E.; Santos, M.; Pandit, R. Wind turbine pitch reinforcement learning control improved by PID regulator and learning observer. *Eng. Appl. Artif. Intell.* **2022**, *111*, 104769. [CrossRef]
45. Bieker, K.; Peitz, S.; Brunton, S.; Kutz, J.; Dellnitz, M. Deep model predictive flow control with limited sensor data and online learning. *Theor. Comput. Fluid Dyn.* **2020**, *34*, 577–591. [CrossRef]
46. Gunther, J.; Pilarski, P.; Helfrich, G.; Shen, H.; Diepold, K. Intelligent laser welding through representation, prediction, and control learning: An architecture with deep neural networks and reinforcement learning. *Mechatronics* **2016**, *34*, 1–11. [CrossRef]
47. Lin, W.S.; Yang, P.C. Adaptive critic motion control design of autonomous wheeled mobile robot by dual heuristic programming. *Automatica* **2008**, *44*, 2716–2723. [CrossRef]
48. Haykin, S. *Neural Networks: A Comprehensive Foundation. 1999*; Mc Millan: Mahwah, NJ, USA, 2010; pp. 1–24.
49. Calise, A.; Rysdyk, R. Nonlinear adaptive flight control using neural networks. *IEEE Control. Syst. Mag.* **1998**, *18*, 14–25. [CrossRef]
50. Kruger, T.; Mossner, M.; Axmann, J.; Vorsmann, P.; Kuhn, A. Adaptive flight control for unmanned aircraft using a stable neural network observer. In Proceedings of the AIAA Infotech at Aerospace 2010, Atlanta, GA, USA, 20–22 April 2010. [CrossRef]
51. Kruger, T.; Schnetter, P.; Placzek, R.; Vorsmann, P. Fault-tolerant nonlinear adaptive flight control using sliding mode online learning. *Neural Netw.* **2012**, *32*, 267–274. [CrossRef]

52. Lin, F.J.; Hung, Y.C.; Chen, S.Y. Field-programmable gate array-based intelligent dynamic sliding-mode control using recurrent wavelet neural network for linear ultrasonic motor. *IET Control. Theory Appl.* **2010**, *4*, 1511–1532. [CrossRef]
53. Zhu, D.; Mei, T.; Luo, M. Adaptive sliding mode control for robots based on fuzzy support vector machines. In Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA 2009), Changchun, China, 9–12 August 2009; pp. 3469–3474. [CrossRef]
54. Wang, Y.; Sun, F.; Liu, J.; Sun, M.; Xie, Y. Application of Smith predictor based on single neural network in cold rolling shape control. *Chin. J. Mech. Eng.* **2009**, *22*, 282–286. [CrossRef]
55. Perez, J.; Cabrera, J.; Castillo, J.; Velasco, J. Bio-inspired spiking neural network for nonlinear systems control. *Neural Netw.* **2018**, *104*, 15–25. [CrossRef]
56. Perez-Peña, F.; Leñero-Bardallo, J.; Linares-Barranco, A.; Chicca, E. Towards bioinspired close-loop local motor control: A simulated approach supporting neuromorphic implementations. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017. pp. 1–4. [CrossRef]
57. Ziegler, J.; Nichlos, N. Optimum settings for automatic controllers. *Trans. ASME* **1942**, *64*, 759–768. [CrossRef]
58. Ziegler, J.; Nichlos, N. Process lags in automatic-control circuits. *Trans. ASME* **1943**, *65*, 433–443.
59. Nishikawa, Y.; Sannomiya, N.; Ohta, T.; Tanaka, H. A method for auto-tuning of PID control parameters. *Automatica* **1984**, *20*, 321–332. [CrossRef]
60. Åström, K.; Hägglund, T. *Advanced PID Control*; ISA: The Research Triangle, NC, USA, 2006.
61. Dong, C.; Tseng, A. A multivariable self-tuning controller for injection molding machines. *Comput. Ind.* **1989**, *13*, 107–122. [CrossRef]
62. Åström, K.; Wittenmark, B. On self tuning regulators. *Automatica* **1973**, *9*, 185–199. [CrossRef]
63. Keshta, H.; Ali, A.; Saied, E.; Bendary, F. Application of Static Var Compensator (SVC) with PI Controller for Grid Integration of Wind Farm Using Harmony Search. *Int. J. Emerg. Electr. Power Syst.* **2016**, *17*, 555–566. [CrossRef]
64. Schillinger, M.; Hartmann, B.; Skalecki, P.; Meister, M.; Nguyen-Tuong, D.; Nelles, O. Safe Active Learning and Safe Bayesian Optimization for Tuning a PI-Controller. *IFAC-PapersOnLine* **2017**, *50*, 5967–5972. [CrossRef]
65. Berkenkamp, F.; Schoellig, A.; Krause, A. Safe controller optimization for quadrotors with Gaussian processes. In Proceedings of the IEEE International Conference Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 491–496.
66. Malikopoulos, A.; Papalambros, P.; Assanis, D. A learning algorithm for optimal internal combustion engine calibration in real time. In Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC2007), Las Vegas, NV, USA, 4–7 September 2008; Volume 6, pp. 91–100. [CrossRef]
67. Etedali, S.; Mollayi, N. Cuckoo Search-Based Least Squares Support Vector Machine Models for Optimum Tuning of Tuned Mass Dampers. *Int. J. Struct. Stab. Dyn.* **2018**, *18*. [CrossRef]
68. Manocha, A.; Sharma, A. Three axis aircraft autopilot control using genetic algorithms: An experimental study. In Proceedings of the IEEE International Advance Computing Confenerce (IACC 2009), Patiala, India, 6–7 March 2009; pp. 171–174. [CrossRef]
69. Liang, D.; Ding, W. Modelling and predicting of a switched reluctance motor drive using radial basis function network-based adaptive fuzzy system. *IET Electr. Power Appl.* **2009**, *3*, 218–230. [CrossRef]
70. Elsodany, N.; Rezeka, S.; Maharem, N. Adaptive PID control of a stepper motor driving a flexible rotor. *Alex. Eng. J.* **2011**, *50*, 127–136. [CrossRef]
71. Mitra, P.; Maulik, S.; Chowdhury, S.; Chowdhury, S. ANFIS based automatic voltage regulator with hybrid learning algorithm. In Proceedings of the Universities Power Engineering Conference, Brighton, UK, 4–6 September 2007; pp. 397–401. [CrossRef]
72. Chen, H.C. Optimal fuzzy PID controller design of an active magnetic bearing system based on adaptive genetic algorithms. In Proceedings of the 7th International Conference on Machine Learning and Cybernetics (ICMLC), Kunming, China, 12–15 July 2008; Volume 4, pp. 2054–2060. [CrossRef]
73. Wen, S.; Hu, X.; Lv, X.; Wang, Z.; Peng, Y. Q-learning trajectory planning based on Takagi–Sugeno fuzzy parallel distributed compensation structure of humanoid manipulator. *Int. J. Adv. Robot. Syst.* **2019**, *16*, 1729881419830204. [CrossRef]
74. Ercin, O.; Coban, R. Identification of linear dynamic systems using the artificial bee colony algorithm. *Turk. J. Electr. Eng. Comput. Sci.* **2012**, *20*, 1175–1188. [CrossRef]
75. Atashpaz Gargari, E.; Hashemzadeh, F.; Rajabioun, R.; Lucas, C. Colonial competitive algorithm: A novel approach for PID controller design in MIMO distillation column process. *Int. J. Intell. Comput. Cybern.* **2008**, *1*, 337–355. [CrossRef]
76. Rahimi, A.; Bavafa, F.; Aghababaei, S.; Khooban, M.; Naghavi, S. The online parameter identification of chaotic behaviour in permanent magnet synchronous motor by Self-Adaptive Learning Bat-inspired algorithm. *Int. J. Electr. Power Energy Syst.* **2016**, *78*, 285–291. [CrossRef]
77. Agand, P.; Shoorehdeli, M.A.; Khaki-Sedigh, A. Adaptive recurrent neural network with Lyapunov stability learning rules for robot dynamic terms identification. *Eng. Appl. Artif. Intell.* **2017**, *65*, 1–11. [CrossRef]
78. Folly, K. Parallel Pbil Applied to Power System Controller Design. *J. Artif. Intell. Soft Comput. Res.* **2013**, *3*. [CrossRef]
79. Thuruthel, T.; Falotico, E.; Renda, F.; Laschi, C. Learning dynamic models for open loop predictive control of soft robotic manipulators. *Bioinspiration Biomimetics* **2017**, *12*, 066003. [CrossRef]
80. Fletcher, R. *Practical Methods of Optimization*, 1st ed.; Wiley: Chichester, UK, 1987.
81. Poyiak, B.T. Introduction to optimization. In *Translations Series in Mathematics and Engineering*; Optimization Software, Publications Division: New York, NY, USA, 1987.

82. Katoch, S.; Chauhan, S.S.; Kumar, V.A. Review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef]
83. Henrichs, E.; Lesch, V.; Straesser, M.; Kounev, S.; Krupitzer, C. A literature review on optimization techniques for adaptation planning in adaptive systems: State of the art and research directions. *Inf. Softw. Technol.* **2022**, *149*, 106940. [CrossRef]
84. Balochian, S.; Baloochian, H. Social mimic optimization algorithm and engineering applications. *Expert Syst. Appl.* **2019**, *134*, 178–191. [CrossRef]
85. Kumar, S.; Kumar, A.; Shankar, G. Crow search algorithm based optimal dynamic performance control of SVC assisted SMIB system. In Proceedings of the 20th National Power Systems Conference (NPSC), Tiruchirappalli, India, 14–16 December 2018. [CrossRef]
86. Abualigah, L. Multi-verse optimizer algorithm: a comprehensive survey of its results, variants, and applications. *Neural Comput. Appl.* **2020**, *32*, 12381–12401. [CrossRef]
87. Sharma, P.; Sharma, K. Fetal state health monitoring using novel Enhanced Binary Bat Algorithm. *Comput. Electr. Eng.* **2022**, *101*, 108035. [CrossRef]
88. Kishnani, M.; Pareek, S.; Gupta, R. Optimal Tuning of PID controller by Cuckoo Search via Lévy flights. In Proceedings of 2014 International Conference on Advances in Engineering & Technology Research (ICAETR 2014), Unnao, Kanpur, India, 1–2 August 2014; pp. 1–5. [CrossRef]
89. Sethi, R.; Panda, S.; Sahoo, B. Cuckoo search algorithm based optimal tuning of PID structured TCSC controller. *Smart Innov. Syst. Technol.* **2015**, *31*, 251–263. [CrossRef]
90. Wang, L.; Liu, Z.; Chen, C.; Zhang, Y.; Lee, S. Support vector machine based optimal control for minimizing energy consumption of biped walking motions. *Intern. J. Precis. Eng. Manuf.* **2012**, *13*, 1975–1981. [CrossRef]
91. Xiao, L.; Zhu, Q.; Li, C.; Cao, Y.; Tan, Y.; Li, L. Application of modified teaching-learning algorithm in coordination optimization of TCSC and SVC. *Commun. Comput. Inf. Sci.* **2014**, *483*, 44–53. [CrossRef]
92. Shouran, M.; Habil, M. Tuning of PID Controller Using Different Optimization Algorithms for Industrial DC Motor. In Proceedings of the 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 4–5 March 2021; pp. 756–759. [CrossRef]
93. Hurwitz, J.; Kirsch, D. *Machine Learning for Dummies*, ibm limited ed.; John Wiley and Sons, Inc.: Hoboken, NJ, USA, 2018.
94. Inoue, T.; Nakano, M.; Iwai, S. High accuracy control of a proton synchrotron magnet power supply. *Ifac Proc. Vol.* **1981**, *14*, 3137–3142. [CrossRef]
95. Tomizuka, M. Zero Phase Error Tracking Algorithm for Digital Control. *J. Dyn. Syst. Meas. Control.* **1987**, *109*, 65–68. [CrossRef]
96. Tomizuka, M. Dealing with periodic disturbances in controls of mechanical systems. *Annu. Rev. Control.* **2008**, *32*, 193–199. [CrossRef]
97. Uchiyama, M. Formation of High-Speed Motion Pattern of a Mechanical Arm by Trial. *Trans. Soc. Instrum. Control. Eng.* **1978**, *14*, 706–712. [CrossRef]
98. Li, W.; Maier, P.; Enge, H. Self-learning control applied to vibration control of a rotating spindle by piezopusher bearings. *Proc. Inst. Mech. Eng. Part I Journal Syst. Control. Eng.* **2004**, *218*, 185–196. [CrossRef]
99. Hladowski, L.; Galkowski, K.; Rogers, E. Further results on dynamic iterative learning control law design using repetitive process stability theory. In Proceedings of the 10th International Workshop on Multidimensional (nD) Systems, (nDS 2017), Zielona Góra, Poland, 13–15 September 2017. [CrossRef]
100. Kurek, J.; Zaremba, M. Iterative Learning Control Synthesis Based on 2-D System Theory. *IEEE Trans. Automat. Contr.* **1993**, *38*, 121–125. [CrossRef]
101. Pakshin, P.; Emelianova, J.; Emelianov, M. Iterative Learning Control Design for Multiagent Systems Based on 2-D Models. *Autom. Remote Control.* **2018**, *79*, 1040–1056. [CrossRef]
102. Seel, T.; Werner, C.; Raisch, J.; Schauer, T. Iterative learning control of a drop foot neuroprosthesis—Generating physiological foot motion in paretic gait by automatic feedback control. *Control. Eng. Pract.* **2016**, *48*, 87–97. [CrossRef]
103. Seel, T.; Werner, C.; Schauer, T. The adaptive drop foot stimulator—Multivariable learning control of foot pitch and roll motion in paretic gait. *Med. Eng. Phys.* **2016**, *38*, 1205–1213. [CrossRef]
104. Mohammadpour, A.; Mishra, S.; Parsa, L. Iterative learning control for fault-tolerance in multi-phase permanent-magnet machines. In Proceedings of the American Control Conference (ACC 2013), Washington, DC, USA, 17–19 June 2013; pp. 5929–5934. [CrossRef]
105. Li, J.; Wang, S.; Wang, J.; Li, J.; Zhao, J.; Ma, L. Iterative learning control for a distributed cloud robot with payload delivery. *Assem. Autom.* **2021**, *41*, 263–273. [CrossRef]
106. Warrier, R.; Devasia, S. Iterative Learning from Novice Human Demonstrations for Output Tracking. *IEEE Trans. Hum. -Mach. Syst.* **2016**, *46*, 510–521. [CrossRef]
107. Silvestri, M.; Pedrazzoli, P.; Boer, C.; Rovere, D. Compensating high precision positioning machine tools by a self learning capable controller. In Proceedings of the 11th International Conference European Society for Precision Engineering and Nanotechnology (EUSPEN 2011), Como, Italy, 23–26 May 2011; Volume 2, pp. 121–124.
108. Kanawaday, A.; Sane, A. Machine learning for predictive maintenance of industrial machines using IoT sensor data. In Proceedings of the IEEE International Conference on Software Engineering and Service Sciences (ICSESS), Beijing, China, 24–26 November 2018; pp. 87–9. [CrossRef]

109. Dai, Y.; Xu, Y.; Dong, Z.; Wong, K.; Zhuang, L. Real-time prediction of event-driven load shedding for frequency stability enhancement of power systems. *IET Gener. Transm. Distrib.* **2012**, *6*, 914–921. [CrossRef]
110. Yixu, S.; Hongbo, L.; Zehong, Y. An adaptive modeling method for a robot belt grinding process. *IEEE/ASME Trans. Mechatronics* **2012**, *17*, 309–317. [CrossRef]
111. Shettigar, A.; Patel, G.; Chate, G.; Vundavilli, P.; Parappagoudar, M. Artificial bee colony, genetic, back propagation and recurrent neural networks for developing intelligent system of turning process. *SN Appl. Sci.* **2020**, *2*. 1–21. [CrossRef]
112. Truby, R.; Santina, C.; Rus, D. Distributed proprioception of 3d configuration in soft, sensorized robots via deep learning. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3299–3306. [CrossRef]
113. Sharaf, S.; Bangert, P.; Fardan, M.; Alqassab, K.; Abubakr, M.; Ahmed, M. Beam pump dynamometer card classification using machine learning. In Proceedings of the SPE Middle East Oil and Gas Show and Conference (MEOS), Manama, Bahrain, 18–21 March 2019. [CrossRef]
114. Shirobokov, M.; Trofimov, S.; Ovchinnikov, M. Survey of machine learning techniques in spacecraft control design. *Acta Astronaut.* **2021**, *186*, 87–97. [CrossRef]
115. Wang, S.; Chaovalitwongse, W.; Babuska, R. Machine learning algorithms in bipedal robot control. *IEEE Trans. Syst. Man Cybern. Part Appl. Rev.* **2012**, *42*, 728–743. [CrossRef]
116. Glavic, M. (Deep) Reinforcement learning for electric power system control and related problems: A short review and perspectives. *Annu. Rev. Control.* **2019**, *48*, 22–35. [CrossRef]
117. Dineva, A.; Mosavi, A.; Ardabili, S.; Vajda, I.; Shamshirband, S.; Rabczuk, T.; Chau, K.W. Review of soft computing models in design and control of rotating electrical machines. *Energies* **2019**, *12*, 1049. [CrossRef]
118. Li, Z.; Adeli, H. Control methodologies for vibration control of smart civil and mechanical structures. *Expert Syst.* **2018**, *35*, e12354. [CrossRef]
119. Di, X.; Shi, R. A survey on autonomous vehicle control in the era of mixed-autonomy: From physics-based to AI-guided driving policy learning. *Transp. Res. Part Emerg. Technol.s* **2021**, *125*, 103008. [CrossRef]
120. Patel, A.; Ramaiya, K.; Bhatia, C.; Shah, H.; Bhavsar, S. Artificial Intelligence: Prospect in Mechanical Engineering Field—A Review. *Lect. Notes Data Eng. Commun.* **2021**, *52*, 267–282. [CrossRef]
121. Mozaffari, A.; Chenouri, S.; Qin, Y.; Khajepour, A. Learning-based vehicle suspension controller design: A review of the state-of-the-art and future research potentials. *eTransportation* **2019**, *2*, 100024. [CrossRef]
122. Lansdell, B.; Kording, K. Towards learning-to-learn. *Curr. Opin. Behav. Sci.* **2019**, *29*, 45–50. [CrossRef]