ISSN 0005-1179 (print), ISSN 1608-3032 (online), Automation and Remote Control, 2026, Vol. 87, No. 4, pp. 359–373.

SPECIAL ISSUE

Hybrid Controllers in Control Problems of Real Objects

A. F. Pashchenko[*,a] **and E. S. Duvanov**[**,b]

[*] *Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia*
[**] *Lipetsk State Technical University, Lipetsk, Russia*
e-mail: [a] *a.paschenko@ipu.ru,* [b] *evgenyduvanov@yandex.ru*

Received July 8, 2025
Revised October 15, 2025
Accepted November 20, 2025

Abstract—The paper continues the authors' research on control of complex technical facilities using hybrid methods. Two approaches are described and analyzed: neural network predictive and fuzzy hybrid controllers. Issues of adaptive tuning, control accuracy, and practical implementation of both methods are considered. Theoretical and practical results of the developed algorithms application in laboratory and industrial experiments are presented.

Keywords: hybrid control methods, neural network predictive controller, fuzzy quadratic regulator, biotechnical control systems, MATLAB

DOI: 10.7868/S1608303226040064

1. INTRODUCTION

Hybrid approaches combining classical algorithms with artificial intelligence methods are increasingly being adopted in modern control systems to enhance adaptability, accuracy, and reliability. Two of the most promising directions in hybrid control are neural network predictive controllers and fuzzy hybrid controllers. The first ones enable prediction of complex facility dynamics and optimization of control actions over a prediction horizon, while the latter effectively adapt to uncertainties and disturbances on the basis of expert rules. Both approaches demonstrate superiority over classical PI controllers in practice, yielding better control quality indices in the respective applied domains.

Artificial neural networks, capable of approximating complex nonlinear dependencies, are widely used for constructing predictive models of controlled devices. Within a conventional scheme, the model is pre-trained on historical data, and after that the neural network predicts future values of the output variable over a specified prediction horizon. Based on these predictions, an optimization algorithm generates the control action, minimizing a selected performance criterion. Thus, in the control system of a complex mechanical facility, a neural network predictive controller had demonstrated higher accuracy and faster response compared to a classical PI controller [1]. The application of a neural network model increases the sensitivity and responsiveness of the system to external disturbances, which is particularly relevant for problems with dynamic alterations. However, such methods require considerable computational resources and substantial training data, and this sophisticates their rapid practical deployment on industrial facilities.

Fuzzy hybrid controllers represent a combination of heuristic expert rules, formalized as membership functions and inference rules, with methods of optimal control. The hybrid architecture combines the robustness of fuzzy logic against uncertainties and nonlinearities with the precision and predictability of classical algorithms, providing adaptive real-time adjustment of control set points and improving the system's response to dynamic disturbances.

Fuzzy hybrid controllers find practical application due to their ability to consider complex interrelationships among various technological parameters. This approach reduces mode oscillations and ensures a smooth, stable transition to the desired set point compared with traditional PI controllers. Hybrid controllers in automatic systems demonstrate significantly higher accuracy in maintaining specified parameters, leading to improved product quality and reduced operational risks [2].

2. DESIGN AND ADJUSTMENT OF THE NEURAL NETWORK PREDICTIVE CONTROLLER

The neural network predictive controller (*NNPC*) is a model predictive control method adapted for nonlinear systems. A neural network trained on experimental data serves as the plant model; at each sampling instant it predicts the dynamics of the output signals. The control error optimization problem is then formulated and solved on the basis of these predictions and the prediction horizon. This approach combines the advantages of model-based predictive control with the flexibility of neural network models. The objective function takes the following form:

$$J = \sum_{i=N_1}^{N_2} \left(y_r(t+j) - y_m(t+j)\right)^2 + \rho \sum_{j=1}^{N_u} \left(u'(t+j-1) - u(t+j-2)\right)^2, \tag{1}$$

where N_1, N_2 are the minimum and maximum prediction horizons (typically $N_1 = 1$), N_u is the control horizon, y_r is the desired reference output, y_m is the network-predicted output, u' is the vector of proposed control inputs, and ρ is the control penalty weighting coefficient.

Figure 1 shows the structure of the control system based on the neural network predictive controller

The network is pre-trained offline in batch mode on experimental data from the actual plant or device, minimizing the prediction error—the difference between the true system output and the network prediction. A two-layer *MLP* (*Multilayer Perceptron*) is typically used for object identification. The network receives as inputs the most recent control and output values of the system (through the lag unit).

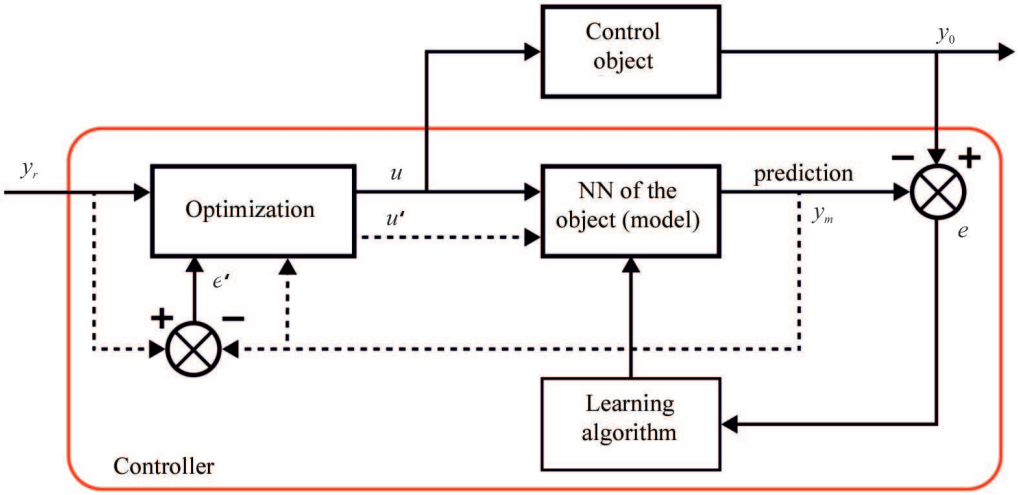

Fig. 1. Structure of the neural network predictive controller-based control system.

The network input vector comprises $u(t), u(t-1), \ldots$ and $y_p(t), y_p(t-1), \ldots$—the current and delayed values of the system's input and output. These inputs are followed by a single hidden layer with a nonlinear (sigmoidal) activation function. The output is a linear combination of the hidden layer, yielding the one-step-ahead prediction $y_m(t+1)$.

This approach rests on the universal approximation theorem [3]: a two-layer neural network can approximate any nonlinear function to sufficient accuracy. After training is complete, the network weights are stored and the model is integrated into the controller for sequential prediction of future system responses. At each time step the network processes input $u'(t)$ and the previous state, producing $y_m(t+1)$; this output is then fed back as an input when necessary to simulate multi-step-ahead behavior depending on the prediction horizon.

Suppose the network is trained so that at time instants $t, t-1, \ldots$, it produces a one-step-ahead prediction $y_m(t+1)$ as given inputs. Repeated iterative application of this network yields predictions $y_m(t+2), y_m(t+3), \ldots, y_m(t+N_2)$, provided that the future inputs $u'(t), u'(t+1)$ are known or assumed.

The first sum in (1) accumulates the squared tracking errors over the horizon $[t+N_1, t+N_2]$; the second penalizes the magnitude of control increments—the difference between the proposed new u' and the last applied u.

When solving the optimization problem, criterion J is minimized with respect to $u'(t), \ldots, u'(t+N_u-1)$; however, in actual control only the first element of the resulting optimal vector is applied.

In the *NNPC* methodology typically $N_1 = 1$ is fixed (tracking error is counted from the very next step). The value $\rho > 0$ is chosen to balance controller responsiveness and control smoothness. The network model is represented as

$$y_m(t+1) = f\big(u(t), u(t-1), \ldots, y_p(t), y_p(t-1), \ldots\big), \tag{2}$$

where $y_p(t)$ is the current object's output. For example, the model is described by

$$y_m(t+1) = W^{(2)}\sigma\big(W^{(1)}[u(t), y_p(t)] + b^{(1)}\big) + b^{(2)}, \tag{3}$$

where $W^{(1)}$ and $W^{(2)}$ are the weight matrices of the first (input) and second (hidden) layers, $b^{(1)}$ and $b^{(2)}$ are the bias vectors, and σ is the sigmoidal activation function [4, 5].

The operation of this controller type is fundamentally based on solving an optimization problem at each sampling instant. The objective function J incorporates two components: the cumulative output tracking error over the prediction horizon and a penalty on the control increment $\Delta u(t)$. As the solution approaches u', the iterative optimization algorithm (Levenberg–Marquardt method) computes the gradient of J with respect to the control parameters and takes a step toward reducing the error [6].

In the practical implementation, at each step the controller performs a finite number of search iterations, after which it adopts and applies the currently found optimal u to the object. By virtue of the iterative scheme and the local approximation of the optimized function, the method converges to a local minimum of J, but does not guarantee a global optimum.

A neural network predictive controller interacting with the controlled device via the *OPC* (*Open Platform Communications*) protocol can be implemented. Figure 2 shows the respective connection diagram. Figure 3 shows the block diagram of the subsystem with the neural network predictive controller.

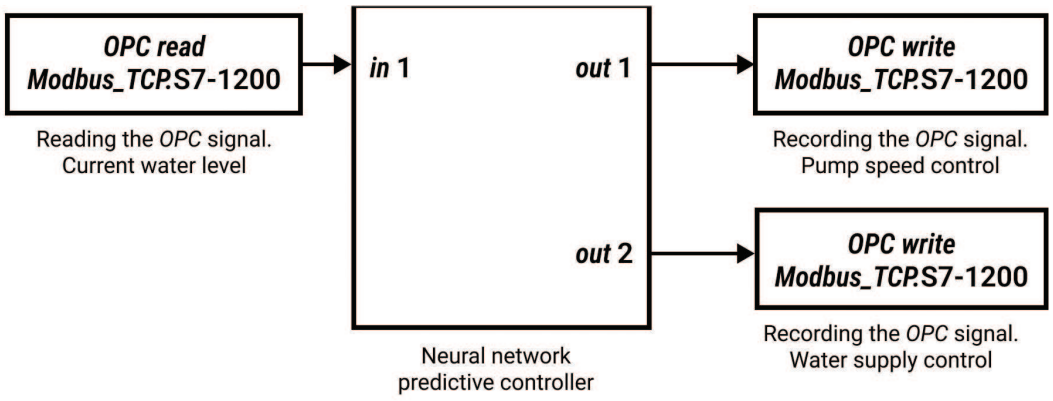

Fig. 2. Connection diagram to the controller via an *OPC* server.

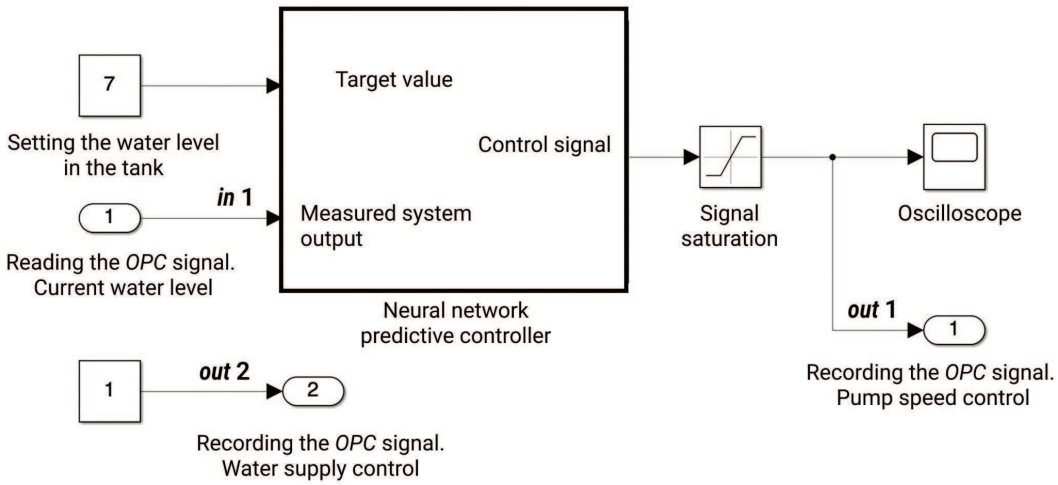

Fig. 3. Subsystem block with the neural network predictive controller.

3. DESIGN AND TUNING OF THE FUZZY QUADRATIC REGULATOR

In 1960, A.M. Letov and R.E. Kalman formulated and successfully solved the linear-quadratic control problem, which consists in finding the optimal control for linear systems using quadratic performance criteria. Letov developed the methodology for time-invariant linear systems, while Kalman extended this approach to non-stationary time-varying systems [7, 8].

The linear quadratic regulator (LQR) implements an optimal control method for linear dynamic systems, based on minimization of the quadratic loss function $e^2(t)$ by use of quadratic criteria. This method aims to minimize a quadratic functional, accounting for the system dynamics and for input and output control parameters, by finding the optimal state feedback gain matrix through the solution of the Riccati differential equation [9].

Consider a linear continuous-time object admitting a vector-matrix state equation representation:

$$\dot{x}(t) = A\,x(t) + B\,u(t), \quad x(t_0) = x_0, \tag{4}$$

where $x \in \mathbb{R}^n$ is the n-dimensional state vector, $A \in \mathbb{R}^{n\times n}$ is the $[n \times n]$ object parameter matrix, $B \in \mathbb{R}^{n\times m}$ is the $[n \times m]$ control matrix with $m \leqslant n$, and $u \in \mathbb{R}^m$ is the control vector.

The state equations of a linear time-invariant system can be written as [9]:

$$\begin{aligned} \dot{x}(t) &= A\,x(t) + B\,u(t), \\ y(t) &= C\,x(t) + D\,u(t), \end{aligned} \tag{5}$$

where A is the $[n \times n]$ system state matrix, B is the $[n \times m]$ input matrix, C is the $[k \times n]$ output matrix, and D is the $[k \times m]$ feedthrough matrix.

The optimality criterion is as follows:

$$J(u(t)) = \min \int_0^\infty [x^T Q\,x + u^T R\,u]\,dt, \tag{6}$$

where matrices Q and R are the weighting matrices for the state variables and control inputs, respectively.

The synthesis objective is to determine the state feedback gain matrix K by minimizing the functional, which reduces to solving the Riccati equation. The Hamiltonian function takes the form

$$H(t, x, u, \lambda) = -\tfrac{1}{2}(x^T Q\,x + u^T R\,u) + \lambda^T (A\,x + B\,u). \tag{7}$$

The gradient conditions can be expressed as

$$\frac{\partial H}{\partial \lambda} = A\,x + B\,u = \dot{x}, \tag{8}$$

$$\frac{\partial H}{\partial x} = A^T \lambda - Q\,x = -\dot{\lambda}, \tag{9}$$

$$\frac{\partial H}{\partial u} = B^T \lambda - R\,u = 0. \tag{10}$$

Using the gradients of the linear form Gz and the quadratic form $z^T S z$, we can write:

$$\frac{\partial}{\partial z}(Gz) = G^T, \qquad \frac{\partial}{\partial z}(z^T S z) = 2\,S\,z,$$

where G is a row vector and S is a symmetric matrix.

From equation (10) it follows that $u = R^{-1} B^T \lambda$. Eliminating the control $u(t)$ from equation (8), we obtain:

$$\dot{x} = A\,x + B\,R^{-1} B^T \lambda, \tag{11}$$

$$\dot{\lambda} = Q\,x + A^T \lambda. \tag{12}$$

To find the optimal control, we solve the system by assuming the existence of a matrix $P(t)$ such that the identity $\lambda(t) = -P(t)\,x(t)$ holds; consequently:

$$u(t) = -R^{-1} B^T P(t)\,x(t). \tag{13}$$

The necessary and sufficient condition for the existence of $P(t)$ is then established, yielding the matrix algebraic Riccati equation for P:

$$PA + A^T P - PBR^{-1} B^T P + Q = 0. \tag{14}$$

Consequently, the linear-quadratic algorithm is a method aimed at minimizing the optimality criterion subject to an equality constraint that incorporates the model of an object. The optimal control law is then set by:

$$u(t) = -K\,x(t), \qquad K = R^{-1} B^T S. \tag{15}$$

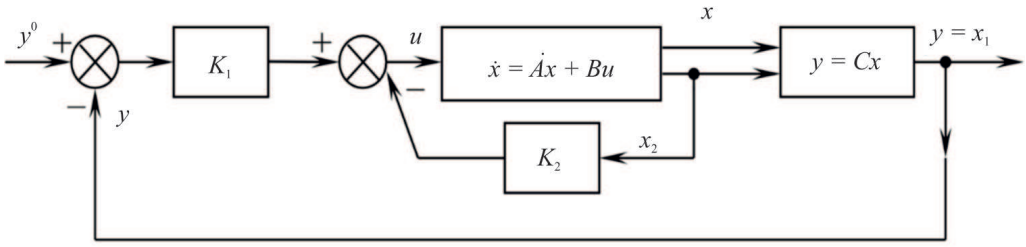

Fig. 4. Control system with the linear-quadratic control algorithm.

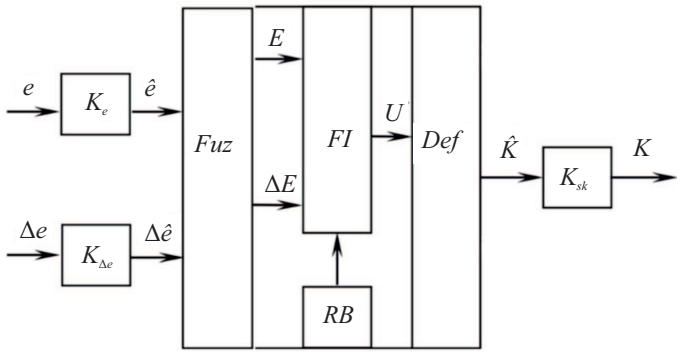

Fig. 5. Structure of the fuzzy quadratic regulator.

It should be noted that the performance criterion can be extended to a finite horizon or to tracking problems (with approximate targeted trajectories); however, in the basic formulation the stabilization-to-equilibrium problem is considered.

The principal advantages of classical LQR, whose schematic representation is shown in Fig. 4, are: guaranteed asymptotic stability of the closed-loop system under full state availability, uniqueness of the solution when the weighting matrices are positive definite, and a systematic (structured) approach to controller synthesis. This control algorithm provides "guaranteed" stability and robustness to small disturbances under an adequate system model. However, the classical LQR has limitations: it requires precise knowledge of the linear object's model and admits only a quadratic problem formulation. In the presence of strong nonlinearities or uncertainties this method may become ineffective; any change in the system structure or parameters requires recomputation of the controller.

To extend the capabilities of the LQR, fuzzy logic-based methods are introduced. The fuzzy quadratic regulator (*FQR*) combines the optimal control law with an adaptive fuzzy inference mechanism [10]. With such approaches the control structure is augmented by a fuzzy logic block that, based on the current system state or tracking error, provides "corrections" to the parameters of the classical controller. For example, a *Linear Fusion Function* may be introduced, combining the LQR output and the fuzzy correction term.

This architecture permits the theoretically grounded properties of optimal control (minimization of the quadratic functional) to be combined with the flexibility of fuzzy rules, providing improved stability, transient response quality, and fault tolerance in the presence of nonlinearities and external disturbances.

In the structure of the nonlinear part of the FQR there are two inputs: the control error $e(k)$ and its rate of change $\Delta e(k) = \Delta e(k)/\Delta t$, and one output K (Fig. 5). Using scaling factors K_e, $K_{\Delta e}$, the actual input values e, Δe are transformed into normalized values $\hat{e}$, $\Delta\hat{e}$ in the interval $[-1, 1]$.

At the fuzzification stage, the normalized input values $\hat{e}$, $\Delta\hat{e}$ are converted into fuzzy variables E, ΔE [1, 11].

The Mamdani fuzzy inference method yields the fuzzy output U from the fuzzy inputs E and ΔE. The defuzzification operation converts the fuzzy output U into a normalized crisp value $\hat{u} \in [L, -L]$ using the centroid method. This normalized value $\hat{u}$ is then multiplied by the scaling factor K_{sk} to obtain the actual value of K.

Adjusting the fuzzy component of the quadratic regulator is a nontrivial task, as it involves both coarse high-level coefficient adjustment and fine low-level tuning to improve control quality.

When constructing a fuzzy model with two inputs and one output, the centroid defuzzification method is typically applied at the defuzzification stage [1]:

$$\hat{y} = \frac{\int_Y y\, Y'(y)\, dy}{\int_Y Y'(y)\, dy}. \tag{16}$$

At the subsequent stage of establishing the input–output relationship of the fuzzy model, the rule base of the fuzzy inference system is applied. Each rule in this base consists of an antecedent and a consequent.

Writing the variables in continuous form: $\hat{e}(t)$, $\Delta\hat{e}(t)$, $u(t)$, $t \in [0, T]$. For a continuous fuzzy controller, the rules take the form:

$$R^\theta\text{: if } \hat{e}(t) \text{ is } E^\theta \text{ and } \Delta\hat{e}(t) \text{ is } \Delta E^\theta, \text{ then } u(t) \text{ is } U^\theta, \tag{17}$$

where E^θ, ΔE^θ, and U^θ are fuzzy sets characterizing the error $\hat{e}(t)$, its rate of change $\Delta\hat{e}(t)$, and the control $u(t)$ on $t \in [0, T]$, belonging to the respective linguistic term sets $E \in T_e$, $\Delta E \in T_{\Delta e}$, $U \in T_u$ with elements that are linguistic values.

Based on the analysis of the obtained transient and phase trajectories, the fuzzy rules for the controller are determined. The construction of the rule base for fuzzy quadratic control involves identifying the necessary linguistic variables for the inputs and output, as well as defining the linguistic terms describing the values of these variables.

It should be noted, however, that if $\hat{e}$ is E_Z and $\Delta\hat{e}$ is DE_Z, the current control value is maintained and K equals K_Z:

$$R\text{: if } \hat{e} \text{ is } E\_Z \text{ and } \Delta\hat{e} \text{ is } DE\_Z, \text{ then } K \text{ is } K\_Z. \tag{18}$$

If the error $\hat{e}$ is not specified, the rules should increase the transient response speed at the largest control error.

In the course of the investigation it was established that the fuzzy logic block output requires actions of the same nature but of different magnitudes for each state variable. Consequently, the fuzzy logic block yields a single coefficient y, while multiple scaling factors are required. The number of these factors depends on the model order. Furthermore, since the obtained model is normalized, the scaling factors must be determined.

Table 1 presents the linguistic variables for the fuzzy quadratic regulator. The abbreviations commonly used to denote linguistic values in fuzzy logic are as follows: *NVB* (*Negative Very Big*), *NB* (*Negative Big*), *NM* (*Negative Medium*), *NS* (*Negative Small*), *Z* (*Zero*), *PS* (*Positive Small*), *PM* (*Positive Medium*), *PB* (*Positive Big*), *PVB* (*Positive Very Big*) [1].

Table 1. Fuzzy Rule Selection for Gain K

$\hat{e}\,/\,\Delta\hat{e}$	NB	NS	Z	PS	PB
NB	NVB	NB	NM	NS	Z
NS	NB	NM	NS	Z	PS
Z	NM	NS	Z	PS	PM
PS	NS	Z	PS	PM	PB
PB	Z	PS	PM	PB	PVB

The fuzzy rules derived from Table 1 are listed below. The selection of these rules is governed by the nature of the control action, the error, and its rate of change:

R^1: if $\hat{e}$ is E_NB and $\Delta\hat{e}$ is DE_NB, then K is K_NVB,
R^2: if $\hat{e}$ is E_NB and $\Delta\hat{e}$ is DE_NS, then K is K_NB,
R^3: if $\hat{e}$ is E_NB and $\Delta\hat{e}$ is DE_Z, then K is K_NM,
R^4: if $\hat{e}$ is E_NB and $\Delta\hat{e}$ is DE_PS, then K is K_NS,
R^5: if $\hat{e}$ is E_NB and $\Delta\hat{e}$ is DE_PB, then K is K_Z,
R^6: if $\hat{e}$ is E_NS and $\Delta\hat{e}$ is DE_NB, then K is K_NB,
R^7: if $\hat{e}$ is E_NS and $\Delta\hat{e}$ is DE_NS, then K is K_NM,
R^8: if $\hat{e}$ is E_NS and $\Delta\hat{e}$ is DE_Z, then K is K_NS,
R^9: if $\hat{e}$ is E_NS and $\Delta\hat{e}$ is DE_PS, then K is K_Z,
R^{10}: if $\hat{e}$ is E_NS and $\Delta\hat{e}$ is DE_PB, then K is K_PS,
R^{11}: if $\hat{e}$ is E_Z and $\Delta\hat{e}$ is DE_NB, then K is K_NM,
R^{12}: if $\hat{e}$ is E_Z and $\Delta\hat{e}$ is DE_NS, then K is K_NS,
R^{13}: if $\hat{e}$ is E_Z and $\Delta\hat{e}$ is DE_Z, then K is K_Z,
R^{14}: if $\hat{e}$ is E_Z and $\Delta\hat{e}$ is DE_PS, then K is K_PS,
R^{15}: if $\hat{e}$ is E_Z and $\Delta\hat{e}$ is DE_PB, then K is K_PM,
R^{16}: if $\hat{e}$ is E_PS and $\Delta\hat{e}$ is DE_NB, then K is K_NS,
R^{17}: if $\hat{e}$ is E_PS and $\Delta\hat{e}$ is DE_NS, then K is K_Z,
R^{18}: if $\hat{e}$ is E_PS and $\Delta\hat{e}$ is DE_Z, then K is K_PS,
R^{19}: if $\hat{e}$ is E_PS and $\Delta\hat{e}$ is DE_PS, then K is K_PM,
R^{20}: if $\hat{e}$ is E_PS and $\Delta\hat{e}$ is DE_PB, then K is K_PB,
R^{21}: if $\hat{e}$ is E_PB and $\Delta\hat{e}$ is DE_NB, then K is K_Z,
R^{22}: if $\hat{e}$ is E_PB and $\Delta\hat{e}$ is DE_NS, then K is K_PS,
R^{23}: if $\hat{e}$ is E_PB and $\Delta\hat{e}$ is DE_Z, then K is K_PM,
R^{24}: if $\hat{e}$ is E_PB and $\Delta\hat{e}$ is DE_PS, then K is K_PB,
R^{25}: if $\hat{e}$ is E_PB and $\Delta\hat{e}$ is DE_PB, then K is K_PVB.

Figure 6 shows the block diagram of the closed-loop control system with the FQR in the MATLAB–Simulink environment.

The set point, accounting for the scaling factor, can be expressed as

$$y'(t) = y^0 K_{y0}, \qquad \text{control error } e(t) = y'(t) - y(t),$$

continuous-to-discrete signal conversion for the fuzzy logic block:

$$e_{ZOH}(t) = \mathrm{ZOH}(e(t), T),$$

differentiation of the discrete signal and rate of change of the error:

$$de(nT) = e_{ZOH}(nT) - e_{ZOH}(nT - T).$$

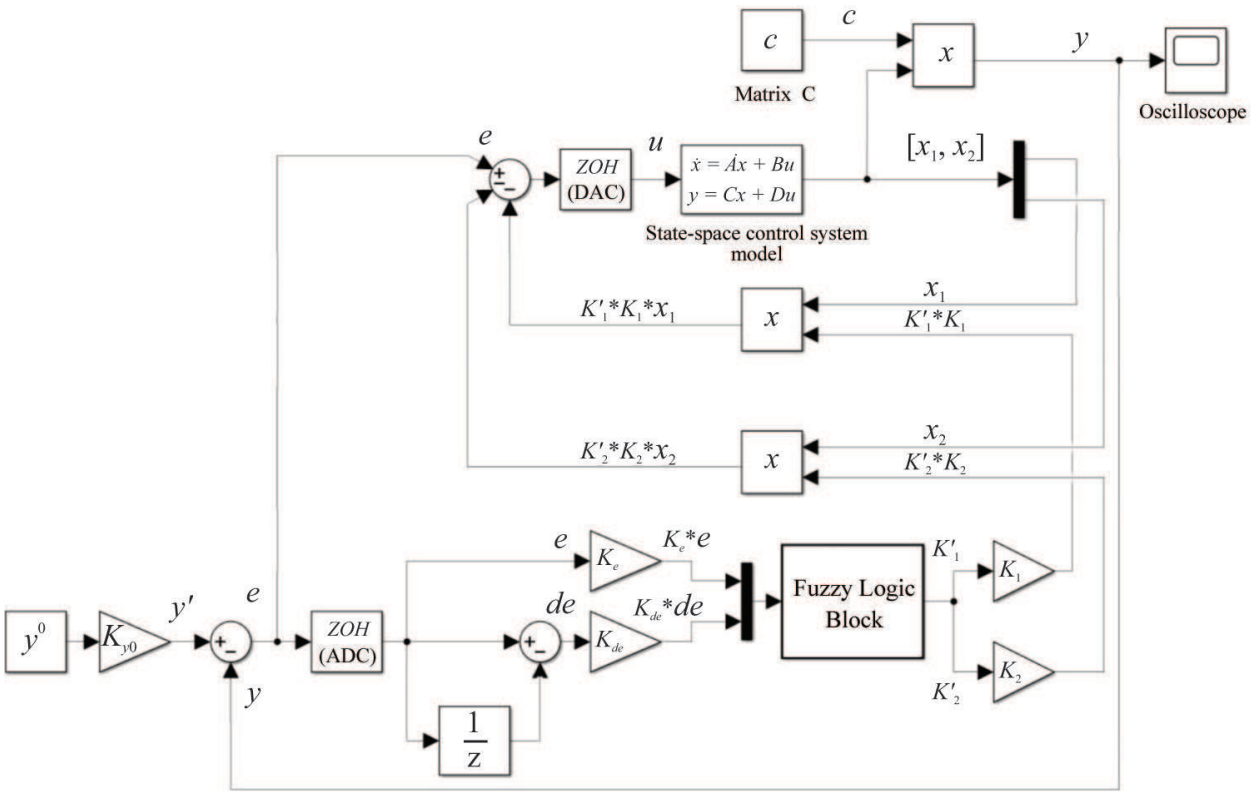

Fig. 6. Block diagram of the closed-loop control system with the fuzzy quadratic regulator.

The fuzzy logic block is described as follows:

$$\begin{bmatrix} K_1' \\ K_2' \end{bmatrix} = \tilde{f}_{\mathrm{FLB}}(K_e\, e_{ZOH},\, K_{\Delta e}\, de). \tag{19}$$

State-space description of the controlled object's model:

$$\dot{X}(t) = A\, X(t) + B\, y(t), \qquad X(t) = C \int_t \dot{X}(t)\, dt.$$

Then

$$X(t) = \begin{bmatrix} X_1(t) \\ X_2(t) \end{bmatrix},$$

consequently,

$$U(t) = e(t) - X_1(t)\, K_1'(t)\, K_1 - X_2(t)\, K_2'(t)\, K_2.$$

Inverse conversion from discrete to continuous signal:

$$u(t) = \mathrm{ZOH}(U(t)), \qquad y(t) = C\, X(t)$$

is performed by the *Zero-Order Hold* block.

In the fuzzy logic block of the quadratic regulator, the gains K_1' and K_2' are adjusted in accordance with the linguistic rules. This gain adaptation procedure is carried out to optimize controller performance, thereby enhancing the efficiency and accuracy of system control.

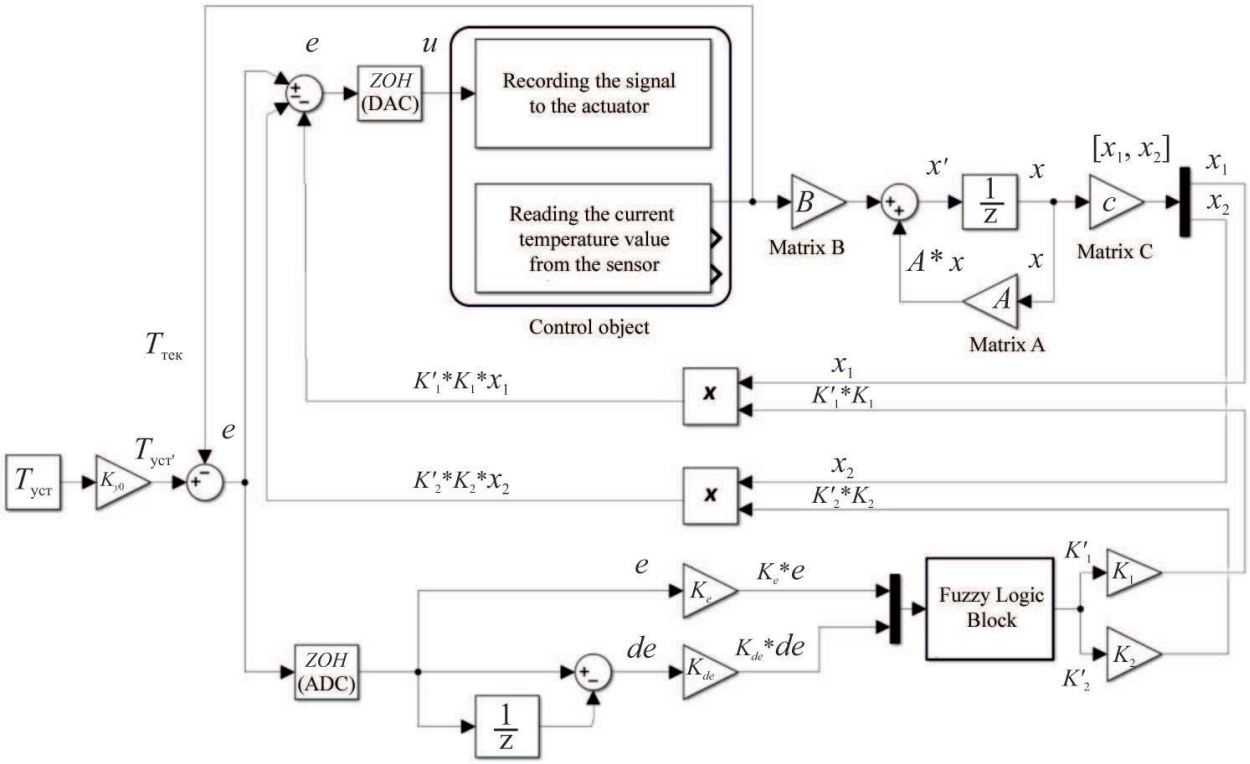

Fig. 7. Block diagram of the closed-loop control system with the fuzzy quadratic regulator and *OPC* server blocks.

Unlike the simulation scheme, in the OPC server scheme when connecting to the controlled device it is necessary to add a state observer block that acquires the object's state vector in real time (Fig. 7).

4. IMPLEMENTATION OF HYBRID CONTROLLERS AT REAL FACILITIES

The paper presents an extended theoretical justification for the application of the neural network predictive controller to control problems involving objects with complex dynamics. It is shown that the use of neural network models enables effective approximation of system behavior over the prediction horizon and, in combination with optimization methods, ensures high control accuracy. The mathematical formalization of the optimality criterion, the implementation of the prediction horizon strategy, and the controller architecture make it possible to consider various constraints, delays, and nonlinear properties of the controlled process.

At present, this type of controller has been successfully applied in research tasks connected with liquid level regulation in a hydraulic plant [12, 13]. MATLAB–Simulink software, a Simatic S7-1200 programmable logic controller, and an OPC server were used in the control implementation.

The MATLAB environment with the Deep Learning Toolbox library enables efficient implementation of complex neural network-based control systems. The *NN Predictive Controller* block (Fig. 3) serves as the built-in NNPC—a controller consisting of the reference input (*reference*), the measured system output input (*feedback*), and the control signal output (*control*).

This block is configured through dialog windows divided into two tabs: *Controller Design* (predictor parameters) and *Plant Identification* (network model parameters). The key block parameters are: prediction and control horizons (N_1, $N_2 = 1$), the weighting coefficient ρ, the optimization

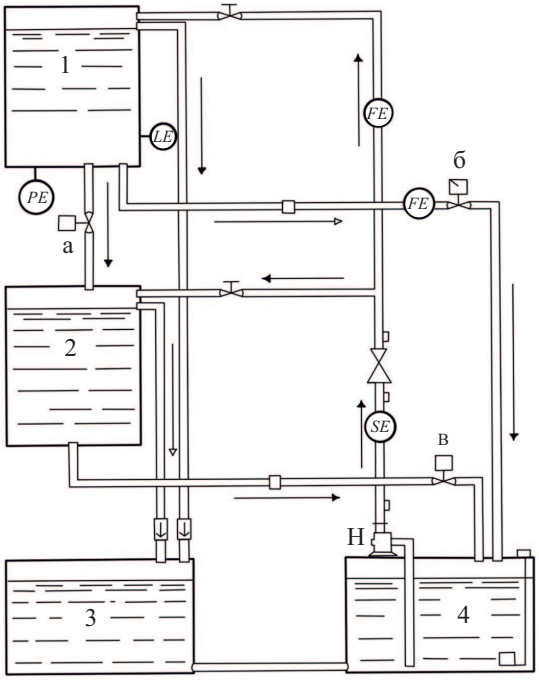

Fig. 8. Research bench "Hydraulic Plant."

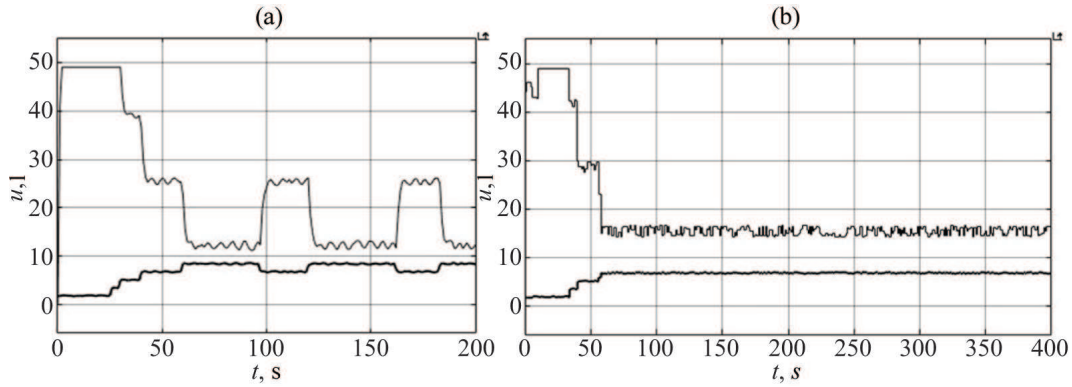

Fig. 9. Transient response using: (a)—classical PI controller, (b)—neural network predictive controller.

parameter α (which determines the required relative reduction of J at each search step), the optimization algorithm, and the number of iterations.

The *Plant Identification* tab specifies the parameters of the trainable object model network: the number of hidden layer neurons, the number of input and output variable delays, the training function, and the training data reading settings.

In the *Simulink Plant Model*, a pre-prepared model with the object's transfer function is selected, from which the neural network obtains a preliminary representation of the object's behavior.

Based on a series of studies, the major objective of which was to maintain the liquid level in tank 2 (Fig. 8) at 7 liters, it is concluded that the level control system based on a classical PI controller is unstable.

This is due to the fact that the parameters of the classical PI controller are unable to compensate for minimal accumulation of excess liquid in the tank, leading to spurious responses to changes (Fig. 9).

The results of the experimental study of the two controller types are presented in Table 2.

Table 2. Experimental Results

Controller type	Rise time, s	Transition time, s	Overshoot, l	Stability behavior
Neural network predictive controller	54	57	0.04	Stable
PI controller	42	66	0.23	Unstable

It should be noted, however, that neural network controllers may be inferior to fuzzy controllers due to their complexity in training and adjustment, high computational cost, and lower predictability of behavior under disturbances and transient conditions.

Fuzzy controllers, by contrast, provide stability and fast response due to simple logical rules, minimal computation time, and transparency of the control algorithm formulation, making them preferable for tasks with precise stability requirements—in particular, in biotechnical control systems where specific process regulations must be strictly observed.

Table 3 presents the types and parameters of the membership functions of the fuzzy component of the quadratic regulator in the *Fuzzy Logic Toolbox* block of MATLAB–Simulink for the temperature regulation task in an incubation cabinet [9, 10, 14]. The fuzzy control rule base was synthesized using the phase plane method.

Table 3. Membership Functions of the Fuzzy Component of the Quadratic Regulator

Fuzzy variable	Membership function	MF type	Parameters
E	E_NB	*trapmf*	[−1.45 −1.05 −1 −0.493]
	E_NS	*trimf*	[−1 −0.5 0]
	E_Z	*trimf*	[−0.5 0 0.5]
	E_PS	*trimf*	[0 0.5 1]
	E_PB	*trapmf*	[0.506 0.999 1.16 1.34]
dE	DE_NB	*trapmf*	[−1.45 −1.05 −1 −0.493]
	DE_NS	*trimf*	[−1 −0.5 0]
	DE_Z	*trimf*	[−0.5 0 0.5]
	DE_PS	*trimf*	[0 0.5 1]
	DE_PB	*trapmf*	[0.506 0.999 1.16 1.34]
K	K_NVB	*trapmf*	[−1.23 −1.02 −0.987 −0.758]
	K_NB	*trimf*	[−1 −0.75 −0.5]
	K_NM	*trimf*	[−0.75 −0.5 −0.25]
	K_NS	*trimf*	[−0.5 −0.25 0]
	K_Z	*trimf*	[−0.25 0 0.25]
	K_PS	*trimf*	[0.00408 0.254 0.504]
	K_PM	*trimf*	[0.25 0.5 0.75]
	K_PB	*trimf*	[0.5 0.75 1]
	K_PVB	*trapmf*	[0.762 0.999 1.03 1.2]

The initial values of gain coefficients K_1 and K_2 (Figs. 6 and 7) are computed using the function $[K, S, e] = \mathrm{lqr}(W\_ss, q, r)$, which provides the baseline controller adjustment with the specified weighting matrices W_ss, q, and r, establishing the foundation for subsequent fuzzy correction according to the defined linguistic rules.

The conversion of the controlled object transfer function to the state-space representation is carried out using the built-in MATLAB function `tf2ss/W_ss`.

Figure 6 illustrates the formal verification procedure for system properties prior to LQR design using the MATLAB functions `ctrb` and `obsv`. First, the controllability matrix

$$C = [B \quad AB \quad A^2B \ \cdots \ A^{n-1}B]$$

is formed; its rank is determined using `ctrb`. If it equals the dimension n of the state vector, the system is declared fully controllable. Similarly, using `obsv` the observability matrix

$$Q = [C^T \ (CA)^T \ (CA^2)^T \ \cdots \ (CA^{N-1})^T]^T$$

is constructed; when it has full rank, all states can be reconstructed, ensuring correct observer operation. In the single-input case, matrix R reduces to a scalar parameter r, which in the optimizing functional

$$J = \min \int_0^\infty (x^T Q\, x + u^T R\, u)\, dt$$

is assigned empirically and plays a key role in balancing control energy consumption and system responsiveness: increasing r reduces the amplitude of the control signal and improves robustness, but slows the dynamic response, whereas decreasing r makes the controller more aggressive, reducing settling time at the expense of increased risk of overshoot and actuator wear.

After the initial computation of the gain coefficients $K = \mathrm{lqr}(A, B, Q, R)$, controllability and observability are verified; then, through successive simulations and experimental tests, the parameter r is adjusted until the required transient response quality indices (settling time, overshoot) are achieved while full controllability and observability are maintained. This iterative optimization serves as an important link connecting the classical LQR approach with the subsequent fuzzy adaptation, enabling high accuracy and reliability in real operating conditions [9, 10].

Thus, the described approach combines rigid optimization via the classical LQR algorithm at the initialization stage with flexible adaptation via fuzzy logic at the real-time stage, ensuring high control accuracy and robustness over a wide range of operating conditions.

The active experiment on temperature regulation in the incubation cabinet was conducted in two stages: finding the optimal control algorithm and controlling a full incubation cycle. The experiments were carried out to study the effect of external factors on the control processes by fully opening the incubation cabinet door for a period of 1200 s, simulating a short-time presence of personnel inside the cabinet. The air temperature in the incubation cabinet for one hour prior to the experiment was 38 °C. The forced-circulation fan was switched on. The results of an experiment are presented in Table 4.

Table 4. Experimental Results for the Incubation Cabinet Temperature Control

Control type	Overshoot, °C	Transition time, s
	Built-in PI controller	
	1.9	3055
	Optimized PI controller	
	1.5	3000
	Fuzzy quadratic regulator	
	1.0	2730

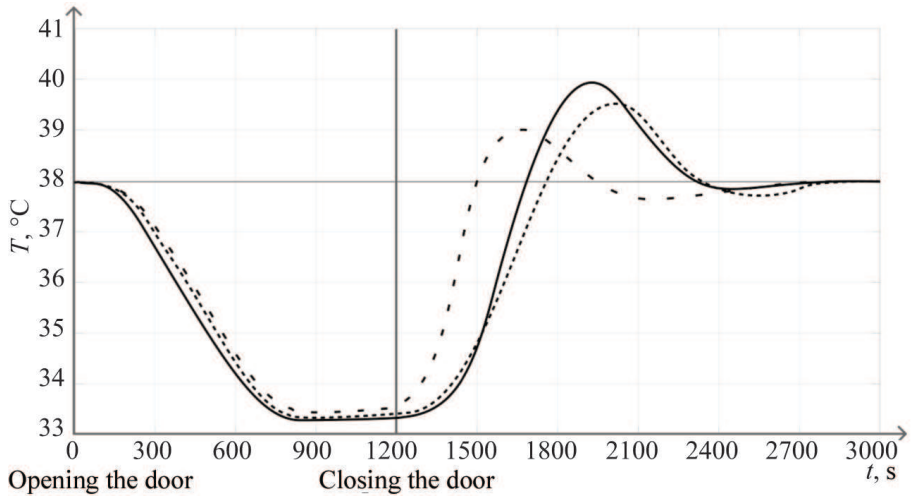

Fig. 10. Effect of external disturbances on the controlled plant in the system with the built-in (solid line), optimized (dense dashed line), and fuzzy quadratic (dashed line) controllers.

Figure 10 shows the response of the controlled object to an external disturbance.

5. CONCLUSION

The choice between a neural network predictive controller and a fuzzy controller should be governed by accuracy requirements, the nature of the object's dynamics, availability of training data and computational resources, and the technological specifics of the application.

Table 5 presents the key decision factors for the selection of a hybrid controller.

Table 5. Key Factors in the Selection of a Hybrid Controller

Characteristic	Neural network predictive controller	Fuzzy controller
Operating principle	Uses an ANN-based plant model; predicts future system behavior and solves the optimization problem via cost functional J.	Control is determined by a set of IF–THEN expert rules; input signals are fuzzified, a fuzzy control action is inferred, and then defuzzified to yield a crisp output value.
Plant model	Identified from data: an ANN is trained to approximate the plant dynamics from experimental data. Capable of representing complex nonlinear dependencies.	No explicit mathematical model; dynamics are defined by rules (described in natural language through linguistic terms).
Interpretability	Low: black-box model; difficult to explain why it produces a particular control action.	High: rules are human-readable and the control logic can be easily modified.
Computational complexity	High: solving an optimization problem in real time is required at each control step. Training can be resource-intensive.	Low: rule evaluation and defuzzification operations are fast; suitable for high-speed controllers.

REFERENCES

1. Pashchenko, F.F., Kudinov, Yu.I., Pashchenko, A.F., Kudinov, I.Yu., Kelina, A.Yu., and Duvanov, E.S., *Nechetkie modeli i systemy upravleniya* (Fuzzy Models and Control Systems), Moscow: Lenand, 2026.
2. Ignatyev, V.V. and Finaev, V.I., The use of hybrid regulator in design of control systems, *World Appl. Sci. J.*, 2013, vol. 23, no. 10, pp. 1291–1297.
3. Cybenko, G.V., Approximation by superpositions of a sigmoidal function, *Math. Control Signals Syst.*, 1989, vol. 2, no. 4, pp. 303–314.
4. Akesson, B.M. and Toivonen, H.T., A neural network model predictive controller, *J. Process Control*, 2006, vol. 16, no. 9, pp. 937–946.
5. Draeger, A., Engell, S., and Ranke, H., Model predictive control using neural networks, *IEEE Control Syst. Mag.*, 1995, vol. 15, no. 5, pp. 61–66.
6. Vasickaninova, A., Bakosova, M., Oravec, J., et al., Neural network predictive controller design, *Chem. Eng. Trans.*, 2017, vol. 61, pp. 121–126.
7. Letov, A.M., Analytical controller design, *Avtom. Telemekh.*, 1960, vol. 21, no. 5, pp. 436–441.
8. Kalman, R., Falb, P., and Arbib, M., *Topics in Mathematical System Theory*, New York: McGraw-Hill, 1969.
9. Kudinov, Y.I., Pashchenko, F.F., Kelina, A.Y., et al., Analysis of control system models with conventional LQR and fuzzy LQR controller, *Procedia Comput. Sci.*, 2019, vol. 150, pp. 737–742.
10. Pashchenko, F.F., Kudinov, Y.I., Pashchenko, A.F., et al., Fuzzy quadratic control of thermal object, *Proc. 2019 1st Int. Conf. on Control Systems, Mathematical Modelling, Automation and Energy Efficiency (SUMMA)*, IEEE, 2019, pp. 288–293.
11. Vassilyev, S.N., Kudinov, Yu.I., Pashchenko, F.F., et al., Intelligent control systems and fuzzy controllers II, *Autom. Remote Control*, 2020, vol. 80, pp. 345–357.
12. Duvanov, E.S., Fedyanin, T.V., and Pashchenko, A.F., The feasibility of using a predictive neural network controller, *Proc. 17th Int. Conf. on Management of Large-Scale System Development (MLSD)*, IEEE, 2024, pp. 1–5.
13. Duvanov, E.S., Batishchev, R.V., Pashchenko, A.F., et al., Applying of predictive neural network controllers, *Proc. 4th Int. Conf. on Technology Enhanced Learning in Higher Education (TELE)*, IEEE, 2024, pp. 422–427.
14. Duvanov, E.S., *Analis i sintez gibridnych regulyatorov dlya prediktivnogo upravleniya teplovym protsessom na osnove nechetkoi logiki* (Analysis and Synthesis of Hybrid Controllers for Predictive Control of a Thermal Process Based on Fuzzy Logic), Extended Abstract of Cand. Sci. (Eng.) Dissertation, Lipetsk: LSTU, 2024.

This paper was recommended for publication by A.A. Galyaev, a member of the Editorial Board