energies

MDPI

Article

Fractional PI-PI$^{\mu}$D Controllers with Neural Network Adaptation in Control System of BLDC Motor Electric Drives

Victor Busher [1], Valeriy Kuznetsov [2,*], Viktor Kovalenko [3,*], Mykola Babyak [4], Valeriy Druzhinin [5,*], Valerii Tytiuk [6], Artur Rojek [2], Kateryna Klochko [7], Ievgen Gurin [3] and Yurii Shramko [8]

1 Department of Electrical Engineering and Electronics, National University "Odessa Maritime Academy", Didrikhson Str., 8, 65052 Odesa, OR, Ukraine; victor.v.bousher@gmail.com
2 Electric Energy Department, Railway Research Institute, 50 Józefa Chłopickiego Street, 04-275 Warsaw, Poland; arojek@ikolej.pl
3 Department of Electrical Engineering and Cyber-Physical Systems, Y. M. Potebnia Engineering Educational and Scientific Institute, Zaporizhzhia National University, 66 Universytetska Street, 69600 Zaporizhzhia, ZR, Ukraine; egurin07@gmail.com
4 Department of Railway Transport, Lviv Polytechnic National University, 12, Stepan Bandera Str., 79013 Lviv, Ukraine; mykola.o.babyak@lpnu.ua
5 Department of Power Engineering, Faculty of Energy, Transport and Management Systems, Non-Profit Joint-Stock Company «Karaganda Industrial University», Republic Avenue, 30, Temirtau 101400, KR, Kazakhstan
6 Department of Electromechanics, Electrotechnical Faculty, Kryvyi Rih National University, Vitaly Matusevich, Str., 11, 50027 Kryvyi Rih, DR, Ukraine; tytiuk@knu.edu.ua
7 Department of Electronics and Electronic Communications, Faculty of Electronics and Computer Engineering, Dniprovsky State Technical University, Dniprobudivska Street, 2, 51918 Kamianske, DR, Ukraine; dstu.cdp@gmail.com
8 Department of Automation, Electrical and Robotic Systems, Faculty of Production Automation and Digital Technologies, Technical University "Metinvest Polytechnic" LLC, Pivdenne Highway, 80, 69008 Zaporizhzhia, ZR, Ukraine; yurii.shramko@mipolytech.education
* Correspondence: vkuznetsov@ikolej.pl (V.K.); victor.l.kovalenko@znu.edu.ua (V.K.); v.druzhinin@tttu.edu.kz (V.D.)

Abstract

This paper investigates, for the first time, the synthesis of a controller that incorporates a fractional-order integral component to achieve a closed-loop astaticism order greater than one. To enhance both static and dynamic accuracy, the controller integrates direct-signal-propagation neural networks within each control channel. The controlled plant is the BLDCM speed loop, which is modeled using a fractional-order differential equation. The study compares the performance of four controller types: a classical PID regulator tuned close to the optimal modulus criterion (IntPID); a fractional PI–PI$^{\mu}$D controller (FrPID) that achieves an astaticism order of at least 1.8; and two hybrid neuro-controllers, NN–IntPID and NN–FrPID. While the FrPID controller reduces the root-mean-square error by nearly a factor of five compared with IntPID, the best results are delivered by NN–FrPID. Specifically, it decreases overshoot eight-fold during a reference step (from 2.98% to 0.35%), lowers the root-mean-square error during linear reference tracking by a factor of eleven, and reduces the relative speed error by more than thirty-five times. When combined with a fast learning algorithm executed at each control-cycle iteration, the controller enables the closed loop to adapt not only to variations in gain coefficients, but also to changes in the fractional-aperiodic order of the plant. These results demonstrate that neural fractional-integral controllers offer strong potential for improving accuracy and robustness in BLDC motor drives and are applicable to a wide range of electromechanical systems.

Keywords: BLDCM; PID regulator; neural network fractional PI$^{\mu}$D regulator

Academic Editor: Chunhua Liu

Received: 31 October 2025
Revised: 18 November 2025
Accepted: 20 November 2025
Published: 23 November 2025

Citation: Busher, V.; Kuznetsov, V.; Kovalenko, V.; Babyak, M.; Druzhinin, V.; Tytiuk, V.; Rojek, A.; Klochko, K.; Gurin, I.; Shramko, Y. Fractional PI-PI$^{\mu}$D Controllers with Neural Network Adaptation in Control System of BLDC Motor Electric Drives. *Energies* **2025**, *18*, 6132. https://doi.org/10.3390/en18236132

1. Introduction

To optimize transient processes in electromechanical systems, an accurate mathematical representation of the controlled plant is essential. In order to apply the full range of analytical methods from linear automatic control theory, system models are typically expressed as sets of ordinary differential equations. However, real control objects are inherently nonlinear. A characteristic feature of electromechanical systems is the presence of power–law relationships in their mathematical descriptions—for example, the dependence of the resisting torque on angular velocity in turbomachinery.

One of the fundamental physical phenomena responsible for the nonlinear behavior of electromechanical energy conversion is magnetic saturation in electrical machines, where the magnetic flux depends nonlinearly on the winding current. This relationship is typically described by a power–law function. DC motors with series excitation, brushless DC motors, and electromagnetic brakes are notable examples of electrical machines that frequently, or almost continuously, operate near the magnetic saturation region [1–3].

Permanent-magnet direct-current motors (BLDC, PMSM) are among the most common types of electrical machines. This popularity is explained by several advantages: a simple mechanical design, high rotational speed (up to 20,000 rpm), relatively low manufacturing cost (compared with permanent-magnet machines), and the ability to operate from both DC and AC power sources. Such universal motors are widely used in household appliances, including washing machines, drills, angle grinders, and vacuum cleaners. In addition, series-excited DC motors provide high starting torque, which makes them suitable for applications such as electric vehicles, elevators, and cranes.

Brushless direct-current motors (BLDCMs) have become increasingly widespread in recent decades. Their production is more resource-efficient: the rotor contains no windings, and the stator requires 30–40% less copper compared with asynchronous motors. The manufacturing process is also simpler. Switched reluctance motors (SRMs) are up to four times cheaper, offer a wider speed range, and can operate across a broader temperature span than permanent-magnet synchronous machines. One reason for these advantages is their salient pole structure. However, this structure also introduces drawbacks: the electromechanical energy conversion occurs in a pulsed mode, generating acoustic noise, and magnetic saturation leads to nonlinear characteristics that complicate control.

These nonlinearities—and their representation in mathematical models by power–law dependences—give rise to characteristic features in the dynamic behavior of electromechanical systems: fast initial transients followed by prolonged settling phases. Similar transient characteristics also appear in systems affected by diffusion-dominated processes and memory effects [4,5]. In such cases, modeling nonlinear plants with integer-order differential equations results in significant errors or forces an unjustified increase in the order of the linear model [6].

Fractional differential equations provide significantly higher identification accuracy for nonlinear electromechanical systems while preserving a relatively simple mathematical structure [2,5–8]. Although fractional calculus has been under development since the late seventeenth century, its most rapid advancement has occurred over the past 30–35 years. This progress is closely linked to the evolution of computing technologies, as numerical computation of fractional operators requires processing large datasets. With the development of various approximation techniques for fractional integrals, controllers incorporating fractional-order components have become applicable even in high-speed systems. This enables improved control quality based on more accurate dynamic models and allows for the optimization of transient behaviors [3,5,9,10].

The main advantage of fractional differential equations over traditional integer-order models with embedded nonlinearities is that they remain within the class of linear differ-

ential equations. As a result, classical automatic control theory tools—such as stability analysis, robustness assessment, and frequency-domain techniques—can be applied to fractional-order systems in a manner analogous to their application to integer-order systems [8,11,12].

In control theory, PID controllers are among the most widely used system components. They consist of proportional, integral, and derivative elements. In its simplest representation, a control loop employing such a controller can be described as two transfer-function blocks—one for the controller and one for the plant—connected through a single negative feedback loop [7,9,12].

The mathematical model of a classical PID controller in the Laplace domain is expressed as a weighted sum of the input signal and its integral and derivative components:

$$H_r(s) = \frac{Y_r(s)}{X(s)} = k_{rp} + k_{ri}\frac{1}{s} + k_{rd}s, \tag{1}$$

where $X(s)$, $Y_r(s)$, k_{rp}, k_{ri}, k_{rd} are input and output signals, coefficients of the proportional, integral and differential components of the controller, respectively.

It is evident that incorporating fractional-order integral and derivative components in (1) enables the design of a more flexible controller, capable of replacing a substantially higher-order integer controller while preserving or even enhancing the quality of both transient and steady-state responses [5,7,8]. Such controllers are denoted by an abbreviation $PI^{\mu}D^{\nu}$ and their transfer function is as follows:

$$H_r(s) = \frac{Y_r(s)}{X(s)} = k_{rp} + k_{ri}\frac{1}{s^{\mu}} + k_{rd}s^{\nu}. \tag{2}$$

If it is possible to use the property of integral and differential calculus $s^{\alpha} = s\frac{1}{s^{1-\alpha}}$, that is, to accept $\nu = 1 - \mu$, then the transfer function of the controller can be written in the following form:

$$H_r(s) = \frac{Y_r(s)}{X(s)} = k_{rp} + \frac{1}{s^{\mu}}(k_{ri} + k_{rd}s), \tag{3}$$

which can significantly simplify the calculation of the output signal of the controller.

The issue of applying fractional-order regulators has been the subject of numerous studies.

In [13], a comparative analysis of different types of controllers was performed, showing that a complex PI-PD controller outperforms traditional and fractional counterparts in terms of accuracy, speed, and noise immunity, but its practical implementation is difficult due to high computational costs. Nevertheless, both complex and fractional controllers demonstrate significant potential for improving the accuracy and robustness of nonlinear system control, which determines the prospects for their further research and optimization.

The results of the study [14] confirm that a complex PI-PD controller has the best accuracy, speed, and interference suppression characteristics, while fractional-order controllers, representing a compromise solution, open up significant prospects for improving the robustness and accuracy of nonlinear object control, which determines the directions for their further study and optimization.

The study conducted in [15] compares the effectiveness of various control strategies–the classic PI controller, its versions optimized using Gray Wolf Optimization (GWO) computational intelligence method and with the help of a deep learning agent with RL-TD3 reinforcement, as well as a fractional-order controller, to improve the performance of a universal power quality compensator (UPQC). The simulation results confirm that the best performance in terms of steady-state error, voltage ripple, and harmonic distortion reduction is provided by control systems based on a reinforcement learning agent (RL-TD3), which outperform classic PI controllers.

The paper [16] presents a comparative study of a fractional-order PID controller (FOPID) and a standard PID controller for a nonlinear two-link manipulator system, where a genetic algorithm was used to tune the controllers. The simulation results, evaluated according to a number of integral criteria (ISE, IAE, ITAE, ITSE) and dynamic characteristics, demonstrated the advantages of the fractional-order controller, and a comparative performance matrix was proposed to facilitate the choice between controllers.

The paper [17] addresses the problem of improving the control accuracy of a nonlinear electromechanical object, namely a valve-inductor motor, using fractional-order PID controllers. Their advantage over similar integer-order controllers is demonstrated.

However, studies [13–17] do not address issues of improving control quality when using ANNs in fractional-order controllers.

However, in [18], a study was conducted on hybrid control systems combining classical and fractional PID controllers with artificial neural networks to solve the problem of precise positioning of a two-link manipulator. The best results in position tracking and resistance to external disturbances were shown by the neural network fractional PID controller (NNFOPID), whose parameters were optimized using the Gorilla Forces Troops Optimization algorithm (GTO).

Research goal. To develop and rigorously evaluate an adaptive control framework for BLDC motor drives based on fractional PI–PI$^{\mu}$D controllers with neural network adaptation, aimed at improving tracking accuracy, enhancing stability margins, and strengthening transient performance under pronounced nonlinearities, parameter variations, and changes in the plant's fractional order.

2. Identification of the Control Object

Accurate modeling of nonlinearities and transient behaviors is essential for analyzing dynamic processes in electromechanical systems. To properly evaluate the performance of the proposed control algorithms under real operating conditions, it is necessary to examine the dynamic characteristics of the controlled object—a brushless direct-current motor.

When subjected to voltage surges, a brushless direct current motor exhibits characteristic transient processes $\omega(t)$, which describe the system's dynamic response to sudden changes in the supply voltage. To analyze these transients and verify the adequacy of the mathematical model, a detailed simulation of the motor was performed in Matlab/Simulink model. The simulation is based on the standard three-phase BLDC motor with Rs = 0.7 Ohm, Ls = 2.7 mH, flux linkage = 0.1194, and J = 0.0027 kg·m^2. The model uses a BLDC motor with four pole pairs available in the Simulink library, ensuring accurate implementation of the electromechanical equations and providing a realistic representation of the motor's dynamic behavior.

This standard model enables a comprehensive analysis of the BLDC motor's operation under various control modes, parameter variations, and load disturbances. The obtained simulation results, illustrating the motor's transient responses in terms of angular speed $\omega(t)$, are presented in Figure 1.

When such transient processes are modeled using an integer-order differential equation, acceptable accuracy can be achieved only if the equation order is at least three. In this case, synthesizing a speed controller that ensures astatic behavior with respect to both reference and disturbance inputs results in at least two derivative terms in the controller structure, which creates practical difficulties for implementation.

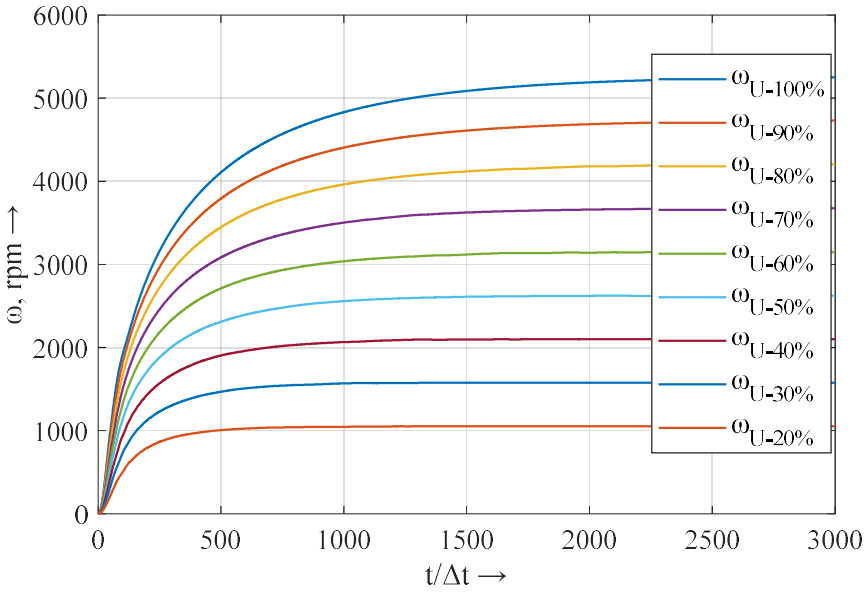


Figure 1. BLDCM start-up graphs at 20, 30, ..., 100% U_n.

When applying fractional-order calculus, powerful tools for system identification are provided by MATLAB R2020a or later toolboxes such as FOMCON, CRONE, and Ninteger [9,11,19–21]. However, they are characterized by a common drawback—they select models with several components of differentiation and integration of different orders, not related to each other by relations of the form $\nu = 1 - \mu$. This also leads to difficulties in the technical implementation of synthesized controllers.

Let us consider the following transfer function as the basic model:

$$H_O(s) = \frac{1}{T_t s^\mu (T_a s + 1)}. \tag{4}$$

When this element is placed under a single feedback loop, the resulting transfer function can be interpreted as a model of the motor, with its output variable normalized to relative units:

$$H_{BLDC}(s) = \frac{\omega(s)}{U_c^*(s)\omega_0} = \frac{1}{T_t T_a s^{1+\mu} + T_t s^\mu + 1}. \tag{5}$$

Considering the uncompensated time constant of the control system T_ν and the gain of the power converter k_p, we obtain the equivalent transfer function of the control object in the following form:

$$H_C(s) = \frac{1}{T_t T_a s^{1+\mu} + T_t s^\mu + 1} \frac{k_p}{T_\nu s + 1}. \tag{6}$$

For different operating speeds, the parameters of this model were determined and can be approximated with sufficient accuracy by linear dependencies on the ideal no-load speed expressed in relative units (Figure 2):

$$\begin{aligned} \mu &= 0.8512 - 0.0450\omega_0^*, \\ T_a &= 0.015 \cdot (0.1954 - 0.1435\omega_0^*), \\ T_t &= 0.015 \cdot (0.3753 + 0.6090\omega_0^*). \end{aligned} \tag{7}$$

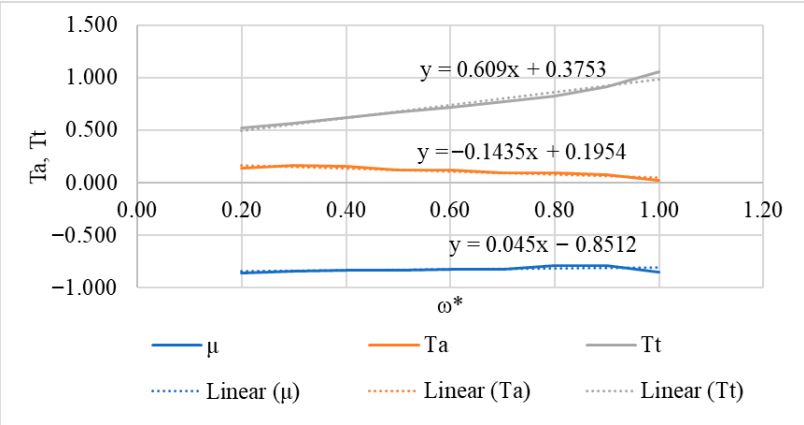


Figure 2. Identification parameters of the BLDCM model at different speeds.

As a result, transient characteristics Y(t) of the motor speed during a voltage surge were obtained, some of which are shown in Figure 3.

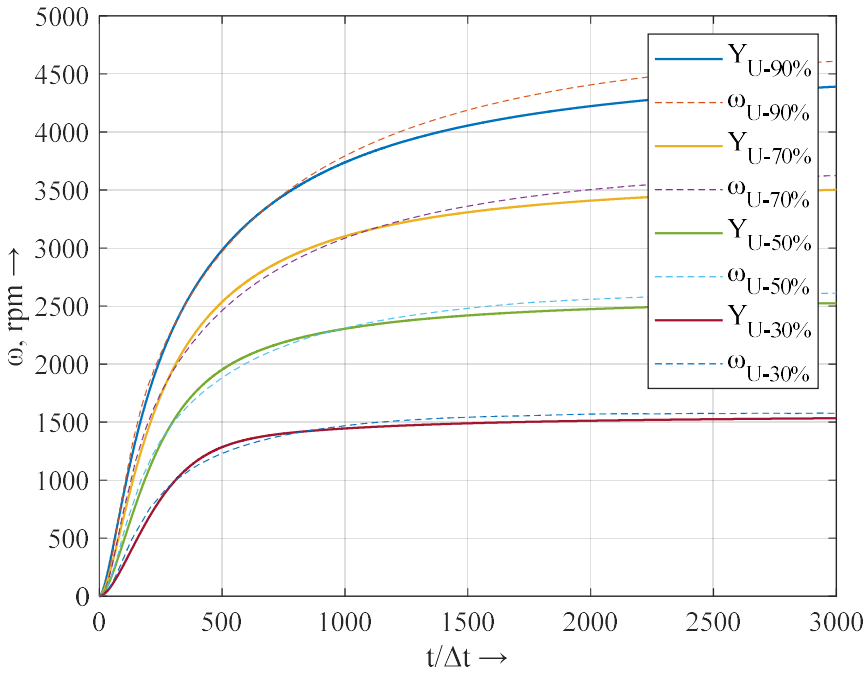


Figure 3. Graphs of transient processes of BLDCM speed, calculated in Matlab by the model based on Formulas (6) and (7).

Table 1 presents the accuracy metrics of the resulting model.

Table 1. Identification accuracy when changing the speed setting.

ω_0, %	20	30	40	50	60	70	80	90	100
RMSE, %	1.897	1.978	1.763	1.675	1.472	1.346	1.194	1.109	1.263

The resulting model provides sufficient static and dynamic accuracy for the synthesis of a closed-loop speed control system.

3. Synthesis of Linear Controllers

Considering the obtained model as a basic one, we synthesize a family of speed controllers to process the given tachograms.

Several assumptions were made in the simulation: the supply voltage limit is set much higher than the nominal one so that the controllers operate mainly in the active mode, not the saturation mode. The motor speed is calculated in relative units, the step Δt is assumed to be the same in all the graphs below, and the relative time $t/\Delta t$ is displayed on the abscissa axes.

The solution of fractional differential equations describing the object is based on the transfer function (4), which allows calculating the fractional integral of the signal only once. Here and earlier, during identification, calculations of transient processes were performed by the finite increment method, using the Oustaloup decomposition [13] with several subranges $N = \pm 8$ from 0 on a logarithmic scale in the frequency range $\omega_L = 0.1^{3.75} \dots \omega_H = 10^{3.75}\ s^{-1}$ for precise coverage of the area 0.01...100 s^{-1}:

$$\begin{gathered} I^{\mu}(s) = \prod_{i=-N}^{N} \frac{s/\omega_{kp\,i} + 1}{s/\omega_{k\,i} + 1}, \\ \lambda = \frac{\omega_H}{\omega_L}, \omega_{kp\,i} = \lambda^{\omega_L(i+N-0.5\mu)/(2N+1)}, \omega_{k\,i} = \lambda^{\omega_L(i+N+0.5\mu)/(2N+1)}. \end{gathered} \tag{8}$$

With such conversion parameters in the frequency range of $0.001 \dots 1000\ s^{-1}$, the deviation of the logarithmic amplitude-frequency characteristic (LAFC) is 0.0129 dB/dec, and the logarithmic phase frequency characteristic (LPFC) is 0.0243 rad, and in the range of interest to us—0.0007 dB/dec and 0.0030 rad, respectively—which ensures the calculation of the fractional integral of a single jump with an error of no more than 0.00080.

Let us consider several options.

The simplest is the classic controller, which can be configured to ensure the modular optimum, assuming that $\mu \approx 1$. For this, the open-loop transfer function, including the controller and the control object, should have the following form [3]:

$$H_{opt}(s) = \frac{1}{aT_{\nu}s} \frac{1}{T_{\nu}s + 1},\ a = 2. \tag{9}$$

Then we obtain a PID controller with parameters:

$$k_{rp} = \frac{T_t}{ak_pT_{\nu}},\ k_{ri} = \frac{1}{ak_pT_{\nu}},\ k_{rd} = \frac{T_aT_t}{ak_pT_{\nu}}. \tag{10}$$

Let us refer to this controller as IntPID. Since it contains an integrating element, placing it under a single negative feedback loop yields an astatic system of order one with respect to both reference and disturbance inputs.

For an object that includes a fractional order element μ, it is possible to synthesize a controller that provides an astatic order $1 + \mu$ with a transfer function of the tuned open loop of the following form:

$$H_{opt}^{1+\mu}(s) = \frac{bT_{\nu}s + 1}{abT_{\nu}^{\mu+1}s^{\mu+1}} \cdot \frac{1}{T_{\nu}s + 1},\ 1 < \mu + 1 < 2,\ b > 1. \tag{11}$$

The synthesis procedure is described in detail in [2,3], and here, we will only use the necessary formulas for calculating the constants a, b:

$$\begin{gathered} (ab) \approx e^{-10.27+7.831(\mu+1)} \Rightarrow \\ b \approx 7.336 + 0.792(ab) + 3.83\ln(ab) \Rightarrow \\ a = \tfrac{(ab)}{b}. \end{gathered} \tag{12}$$

Then,

$$\begin{aligned} &H_C = \frac{k_p}{(T_v s+1)\left(T_a T_t s^{\mu+1}+T_t s^{\mu}+1\right)} \Rightarrow \\ &H_{PI-PI^{\mu}D}(s) = H_{opt}^{1+\mu}(s)/H_C(s) = \\ &= \left(1+\frac{1}{bT_v s}\right)\left(\frac{T_a T_t s}{aT_v^{\mu}k_p}+\frac{T_t}{k_p a T_v^{\mu}}+\frac{1}{k_p a T_v^{\mu} s^{\mu}}\right). \end{aligned} \tag{13}$$

$PI-PI^{\mu}D-regulator$ is obtained. Let us call such a controller FrPID.

It contains only one component with a fractional order of integration, the value of which U_{OUT} for the input signal $U_{IN} = e = Uref - Yc$ can be calculated by the finite increment method, using the following formula for the factors in (8) for each j-th moment of time:

$$\begin{aligned} &x_{1\,j} = U_{IN\,j}; \\ &\Delta x_{i\,j} = x_{i\,j} - x_{i\,j-1}; \\ &y_{i\,j} = 0.5\left(\begin{array}{l} y_{i\,j-1}(1-\omega_{k\,i}\Delta t) + \left(\Delta x_{i\,j}/\omega_{kp\,i} + x_{i\,j}\Delta t\right)\omega_{k\,i} + \\ +\left(y_{i\,j-1} + \left(\Delta x_{i\,j}/\omega_{kp\,i} + x_{i\,j}\Delta t\right)\omega_{k\,i}\right)/(1+\omega_{k\,i}\Delta t) \end{array}\right); \\ &x_{i+1\,j} = y_{i\,j}; \\ &U_{OUT\,j} = I^{\mu}(U_{IN}) = y_{2N+1j}. \end{aligned} \tag{14}$$

This procedure can be readily adapted to a microcontroller programming language and implemented directly into the control software.

A notable feature of the synthesized controller is that it provides the system with an astaticism order greater than one while maintaining stability margins comparable to those achieved with modulus-optimal tuning. As a result, the system exhibits reduced oscillations and eliminates not only the steady-state error but also the speed tracking error.

Let us examine a system with different types of controllers. From here on, the control object will be the BLDC motor, accelerating to its nominal speed under the same conditions considered when identifying this motor at the start of operation.

As a result of simulating a system with the two controller types under a linearly varying reference signal, the transient response plots shown in Figure 4a were obtained. Visually, the responses appear almost identical for both controllers. However, Figure 4b clearly demonstrates that the error trajectories differ significantly: the root-mean-square error over the initial segment of the transient process (from point 1 to 10,000) and over the full interval (from point 1 to 50,000) is nearly five times lower.

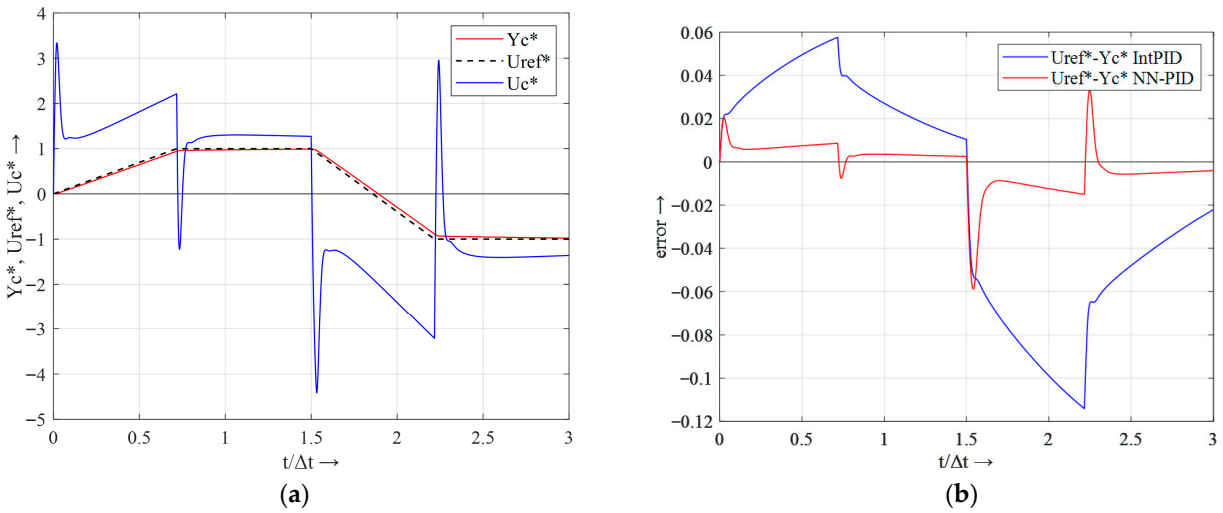


Figure 4. Transient processes graphs of the reference signal Uref*, converter voltage Uc*, speed Yc* (**a**) and error (**b**) when using linear controllers.

Table 2 shows the root-mean-square errors characterizing the beginning (1:10,000) and the entire transient process (1:50,000).

Table 2. Root-mean-square errors when using IntPID, FrPID.

Controller	RMSE (1:10,000)	RMSE (1:50,000)
IntPID	0.0102	0.0107
FrPID	0.0021	0.0027

Table 2 and Figure 4 show the advantages of the fractional-integral controller: it not only reduced the mean square error by five times, but also ensured a constant or even decreasing speed error with a linear change in the reference signal, and in the steady-state mode the error is 5–10 times smaller and tends to 0.

4. Synthesis of Neurocontrollers

The synthesized linear controllers provided reasonably satisfactory results. However, variations in plant parameters at different operating speeds—and potentially under the influence of additional external or internal factors such as temperature—may adversely affect control accuracy.

One possible approach to achieving adaptability is the use of neural controllers [22–25], whose structure is identical to that of the synthesized linear IntPID and FrPID controllers. In these controllers, the fixed-gain amplifiers are replaced by relatively simple fully connected feedforward neural networks with a small number of neurons (Figure 5).

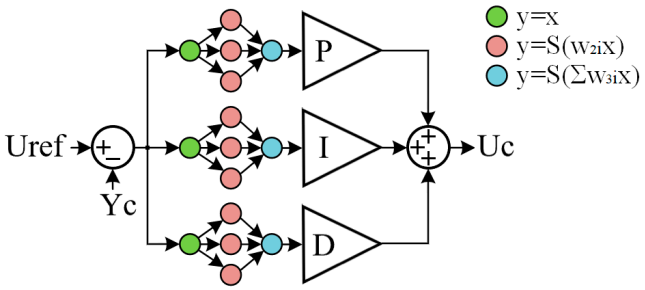


Figure 5. Functional diagram of the neurocontroller.

Let us designate such controllers as NN-IntPID, NN-FrPID, respectively.

Then we can conditionally (mindful about the incorrectness of using the nonlinear functions in the Laplace transform) write the transfer functions of the controllers in the following form:

$$\begin{aligned} H_{NN-PID} &= NN_P \frac{T_t}{ak_pT_v} + NN_I \frac{1}{ak_pT_vs} + NN_D \frac{T_aT_ts}{ak_pT_v}, \\ H_{NN-FrPID} &= \left(1 + \frac{1}{bT_vs}\right)\left(NN_P \frac{T_t}{k_paT_v^\mu} + NN_I \frac{1}{k_paT_v^\mu s^\mu} + NN_D \frac{T_aT_ts}{aT_v^\mu k_p}\right), \end{aligned} \tag{15}$$

where NN_P, NN_I, NN_D are nonlinear functions of neural networks that complement the corresponding components of PID controllers.

Such neural networks should contain one neuron in the input layer, one in the output layer, and 5–10 neurons in the hidden layer. If hyperbolic tangent is used as activation functions, then they form some centrally symmetric nonlinear dependence between the magnitude of error and the output values of individual components. Such three-layer ANNs are generated as sets of matrices of a given configuration with random weight coefficients of neurons of the hidden layer w2 and output neurons w3. The following are

fragments of the program in the language used in the Matlab m-files; however, they can be easily adapted to almost any other programming language that is intended for working with matrices:

```
% initialization
w2 = rand(1,10) * 3.5;
w3 = rand(10,1) * 3.5.
```

Traditionally, the mean square error is calculated on a certain test interval with subsequent correction of the network weight coefficients by the backpropagation method using the Levenberg–Marquardt or gradient descent algorithms for training ANNs [25–27]. However, for the considered class of objects, a fast-training algorithm can be applied during each cycle of executing the transient process calculation program [28]. For this purpose, the model of the control object is used, and the error is calculated in each cycle of calculating the controller output signal and the model response $e = Uref - Yc$. Since the synthesized controller is designed to achieve an error equal to zero, its value is used to correct the weight coefficients by the backpropagation method. We will also take into account that when using a hyperbolic tangent as the activation function of the hidden layer neurons and the output neuron $S(x) = \tanh(x)$, the gradient descent method is based on a simple relationship $S\prime(x) \sim \left(1 - S(x)^2\right)$. Then, the calculation of the correction value of the weight coefficients is carried out according to the formula [29]:

$$\frac{\partial E}{\partial w_{jk}} = -\alpha(t_k - e_k)\left(1 - \tanh^2\left(\sum_j w_{jk} \cdot e_j\right)\right) \cdot e_j, \tag{16}$$

where E—round mean square error, $t_k - e_k$—error of output neurons, e_j—error of hidden neurons, the learning coefficient α is set in the range from 0.1 to 1.0 and, understanding the matrix processing procedures as the operators «*»—vector product, «.*»—element-wise multiplication, and «'»—matrix transposition. Based on this formula, the core of the program is obtained:

```
% regulator
e = Uref − Yc;
Uc = Kr * S(S(e * w2) * w3);

% object in finite increments
Yc = f(Uc, dt, . . .);

% learning–error back-propagation with scaled conjugate gradient for w2, w3
e2 = w3' * e;
s3 = tanh(e2);
S3 = (1 + s3) .* (1 − s3);
dE3 = e .* (S3 .* s3)';
w3 = w3 + alfa * dE3;
e1 = w2' * e2;
s2 = tanh(e1);
S2 = (1 + s2) .* (1 − s2);
dE2 = e2 .* (S2 .* s2)';
w2 = w2 + alfa * dE2;
```

It is important to note that it is advisable to conduct training with a changing reference signal so that the controller does not enter saturation mode. We chose a signal linearly increasing from 0 to 1.

As a result of repeating this procedure for 100–200 epochs, we obtained the settings of the controller, which we then tested by applying a jump and a linearly changing signal.

The changes in the system's behavior with NN-IntPID during training are shown in Figure 6 (one can see how the error gradually decreases during the learning process, especially when changing the derivative of the input signal), and the response to testing the signals after training is shown in Figure 7.

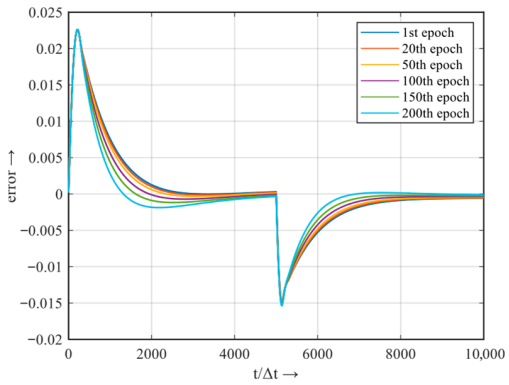
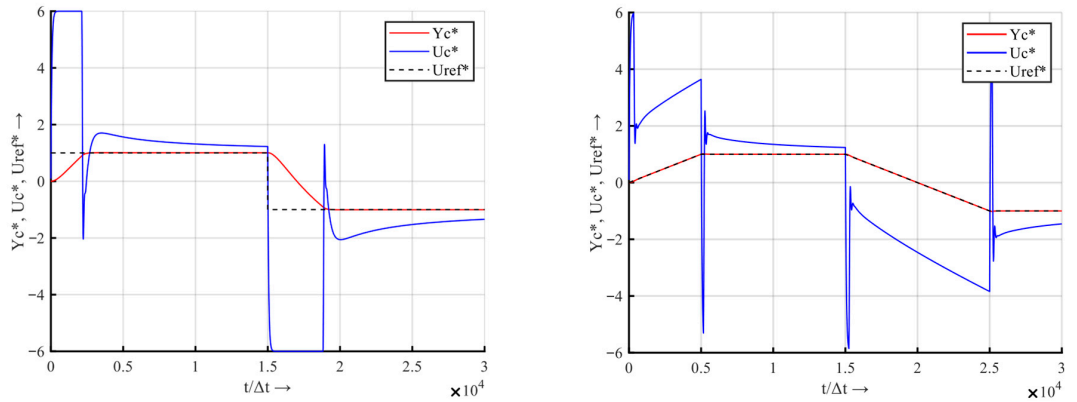


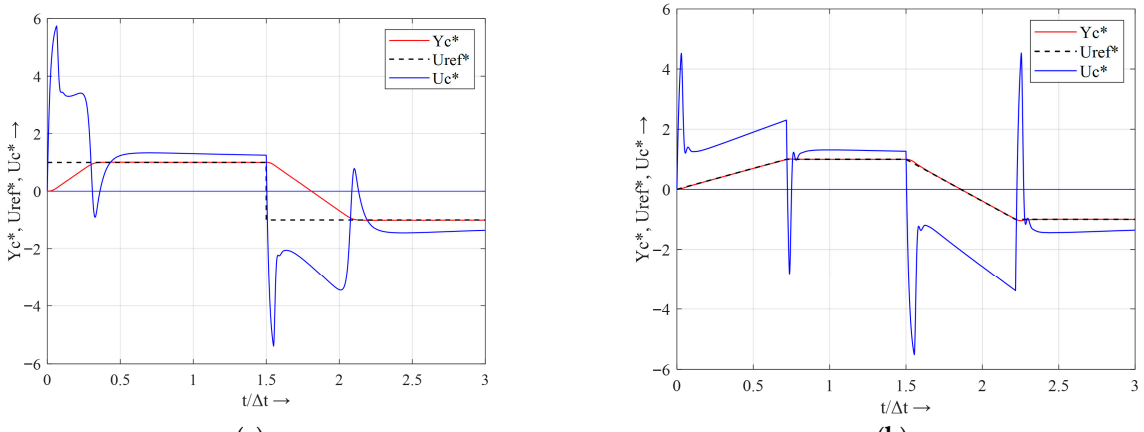
Figure 6. Changes in the behavior of the object with NN–IntPID during training.



(**a**) (**b**)

Figure 7. Response of the trained system with NN–IntPID to jumps (**a**) and linear changes (**b**) of the reference signal.

A similar procedure was carried out with the NN–FrPID controller (Figure 8).



(**a**) (**b**)

Figure 8. Response of the trained system with NN–FrPID to jumps (**a**) and linear changes (**b**) in the reference signal.

Figure 9 compares the graphs of the change in the error in transient processes when using different controllers, and Table 3 shows the change in the relative root mean square error depending on the controller type.

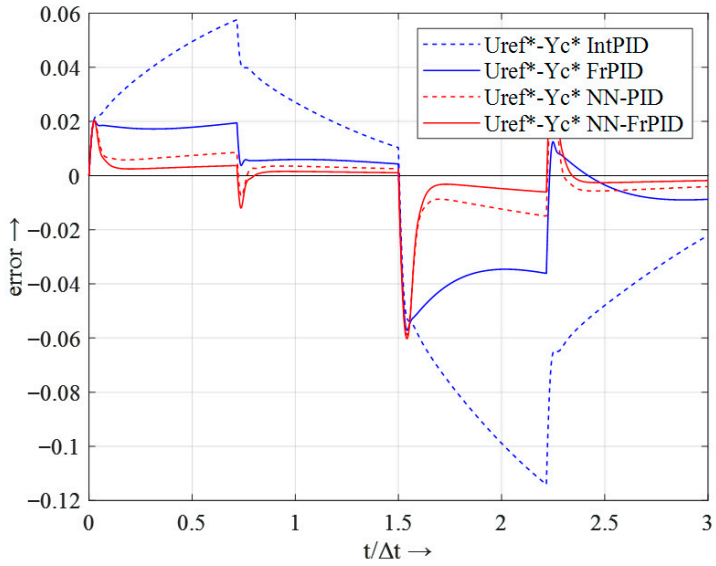


Figure 9. Graphs of transient processes $e = Uref - Yc$ using different controllers.

Table 3. Comparative indicators $e = Uref - Yc$ of IntPID, FrPID and NN IntPID, NN FrPID.

Controller	RMSE (1:10,000)	RMSE (1:50,000)
IntPID	0.0102	0.0107
FrPID	0.0021	0.0027
NN–IntPID	0.0016	0.0022
NN–FrPID	0.0007	0.0016

It is important to note that using the parameters of the IntPID and FrPID controllers as a preliminary setting allows us to find the required parameters of the neurocontroller faster, even if the parameters of the object differ from the calculated ones. Transient processes were obtained with significant differences between the "calculated" and "actual" parameters (T_a, T_t is 25% higher, T_v is twice less; T_a, T_t are 25% lower, T_v is twice more); however, in both cases, the training result was achieved—during acceleration, RMSE was 0.00089 and 0.00071, respectively.

Additionally, the developed controllers successfully operate in a system with not only time constants that have changed or are changing, but also the order of the fractional differential equation, which may correspond to changes in the properties of the BLDC motor at different speeds and loads. Thus, in a system configured with an object described by a fractional differential equation with order $\mu = 0.8$, testing was carried out with a changing order during the transient process from 0.5 to 0.8 and from 0.8 to 0.5. In both cases, NN–FrPID provides the best accuracy (Table 4).

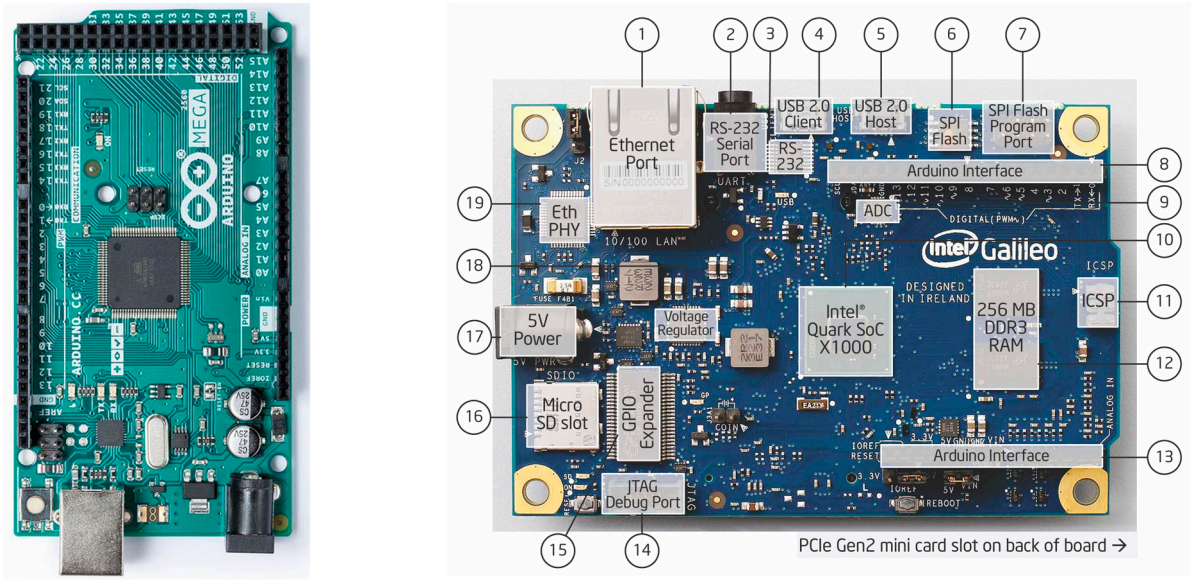
Table 4. Comparative indicators of IntPID, FrPID and NN-IntPID, NN-FrPID when working with an object with a changing order μ.

Controller	$\mu = -0.5 - 0.3t/t_{tp}$		$\mu = -0.8 + 0.3t/t_{tp}$	
	RMSE (1:10,000)	RMSE (1:50,000)	RMSE (1:10,000)	RMSE (1:50,000)
IntPID	0.0079	0.0107	0.0119	0.0107
FrPID	0.0014	0.0027	0.0024	0.0027
NN–IntPID	0.0011	0.0020	0.0018	0.0020
NN–FrPID	0.0008	0.0017	0.0007	0.0017

5. Experimental Implementation of the Controller Based on the Intel Galileo Gen. 2 Microcontroller

Currently, examples of control systems with fractional-integral controllers are known, for example, [2,13–15], along with their advantages over classical PID controllers [2,16,17]. Various methods for approximately calculating the fractional integral of a time-varying signal are also known. The procedure proposed in this paper for optimizing the parameters of these controllers using artificial neural networks requires verification of the feasibility of calculating the fractional integral and procedures related to the operation and training of feedforward neural networks based on single-chip microcontrollers used in industrial devices.

The main operation that the controller must perform is the calculation of the fractional integral. Therefore, to verify the feasibility of implementing the proposed regulator based on Formulas (8) and (14), a program was developed for Arduino series controllers: Mega2560 and Intel Galileo Gen. 2 (Figure 10). The program code is provided in the Addendum.



(**a**) (**b**)

Figure 10. Arduino Mega 2560 (**a**) and Intel Galileo (**b**) controllers (photos from Arduino.com).

The first controller is based on a 16-bit ATmega328 (ATMEGA2560) processor from Microchip Technology (Chandler, AZ, USA) with integer arithmetic. The Arduino IDE environment provides the necessary libraries for handling standard C++ data types and performing mathematical operations with real numbers. However, such operations are executed through separate subroutines, and testing showed that for calculating a fractional

integral of order 0.5 with a time step of 0.01 s, this processor requires an approximately MC longer computation time—2.17 to 2.39 s.

The second controller is based on a 32-bit Intel® Quark™ SoC X1000 processor from Intel Corporation (Santa Clara, CA, USA) equipped with a mathematical coprocessor, including support for real numbers in double format. For this processor, performing fractional-integral calculations using the Oustaloup transformation over the frequency range of $0.1^{3.75}\ldots10^{3.75}$ with a time step of 0.01 s takes about twice the required computation time. However, reducing the frequency range to $0.1^{2.75}\ldots10^{2.75}$ s^{-1} significantly accelerates the calculations—the program execution time for an actual interval of 2 s was 0.407 s, meaning that the processor achieves a speed five times greater than the minimum required (Appendix A. Sketch_fractional_integral.ino). Under these frequency range conditions, between 0.01 and 100 s^{-1}, the deviation of the LAFC is 0.0160 dB/dec, and of the LPFC is 0.0095 rad (vs. 0.0113 dB/dec and 0.0030 rad), the root-mean-square error increases to 0.0089 (compared to 0.00080), and the graph of the instantaneous error, which increases by approximately seven times, is shown in Figure 11.

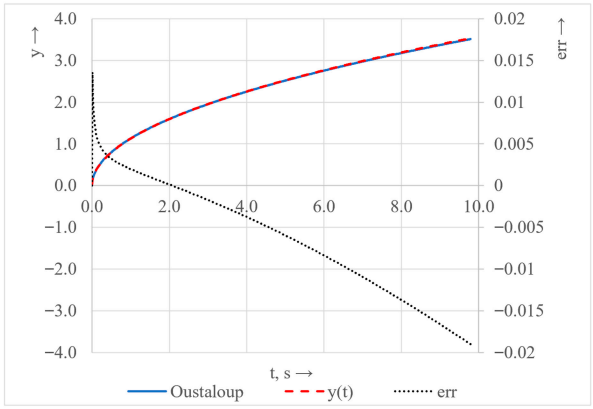


Figure 11. Exact and approximate fractional integral of order 0.5 with the Oustaloup transformation frequency range of $0.1^{2.75}\ldots10^{2.75}$ s^{-1}.

Using the Serial Plot tool, a graph of the output signal from the program calculating the fractional integral of order 0.5 under a unit step input was constructed. The calculation in the program is performed with the time step Δt required to achieve the desired accuracy, while the display step in Serial Plot is chosen according to the number of points stored in this tool, 50 (Figure 12).

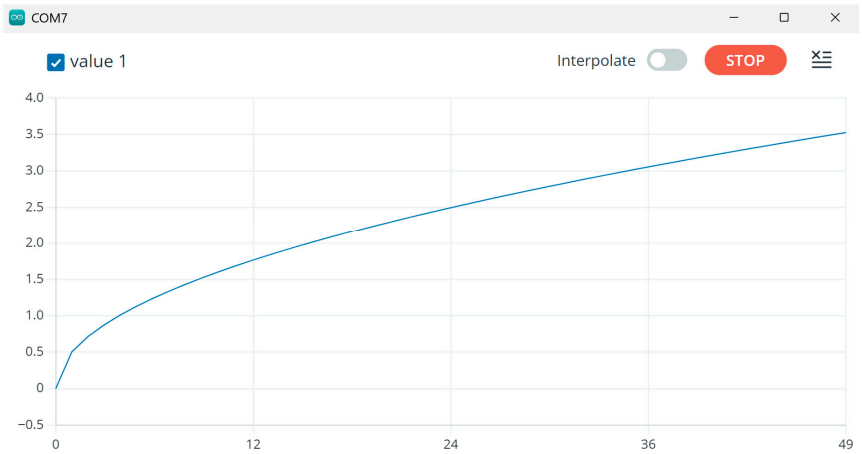


Figure 12. Result of calculating the fractional integral of order 0.5 in the Arduino Intel Galileo Gen. 2 controller over the interval 0. . .9.8 s with a display step of 0.2 s.

It is evident that the graph coincides with the one obtained using computer-based calculations. For engineering computations in feedback control systems, this level of error is considered acceptable.

The neural network training procedure (Appendix A. Sketch_ANN_learning.ino) is performed significantly faster (Figure 13): we deliberately increased the number of epochs to 1000 and 10,000 and obtained the following average computation times for 10 neurons in the hidden layer—61. . .62 ms/1000 epochs and 599. . .601 ms/10,000 epochs, i.e., approximately 0.06 ms per epoch; for 3 neurons—0.02 ms per epoch; and for 2 neurons—0.014 ms per epoch.

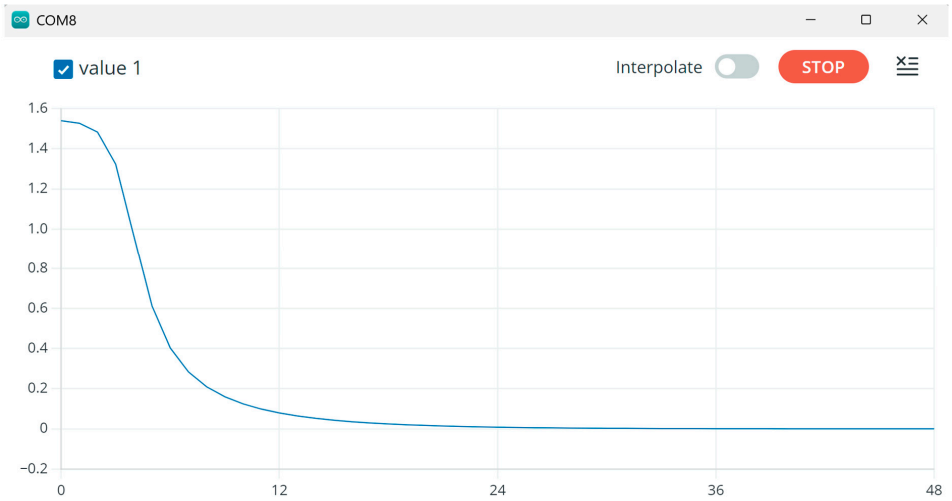


Figure 13. Error variation during the neural network training process in the Intel Galileo Gen. 2 controller.

Thus, it can be concluded that the feasibility of implementing the proposed controllers is well supported.

6. Discussion and Conclusions

In this study, the start-up processes of a brushless direct-current motor were modeled, and it was demonstrated that, with an error not exceeding 2%, the resulting transient responses can be described as the solution to a fractional differential equation, with the plant characterized by the corresponding transfer function (6). Using a classical PI or PID controller in the control system of such a plant yields a relative error under a linear reference signal of approximately 0.010–0.012; however, the speed error increases to about 0.025, and the static error only decreases slowly, which is a behavior characteristic of fractional-aperiodic systems. A substantial improvement—namely a fivefold or greater reduction in the root-mean-square error, along with a rapid decrease in both the static error and the relative speed error to 0.002—is achieved when employing a controller with fractional-order integral and derivative components, which imparts to the system an astatic behavior of order 1.8 with respect to reference and disturbance inputs.

Nevertheless, even these performance levels can be further enhanced. Higher control accuracy was obtained through a synergistic approach combining two areas within the field of soft computing–neural network control and fractional differential and integral computation. A neurocontroller with the same structural configuration as the FrPID, but augmented with fully connected three-layer feedforward neural networks acting as preamplifiers for the fractional integral and proportional components, enables both the static error and the speed error to approach zero over most of the transient response.

In our opinion, this result became possible due to two more factors: 1. using a fast training algorithm—in each cycle of the transient process calculation program—based only

on the instantaneous error value and considering that the aim of training is to reduce the error to zero; 2. preliminary selection of the structure and coefficients of the controller, which allows you to quickly find the necessary weighting coefficients of the neurocontroller, even if the actual parameters of the object differ from the calculated ones. The check has shown that not only deviations of the time constants, but also a change in the order of the fractional differential equation in the description of the object do not lead to instability, oscillation or any significant changes in the dynamic and static indicators, as evidenced by a comparison of the error graphs in Figure 10 and the root-mean-square errors given in Tables 3 and 4.

It is also important to note that, given the similarity in the mathematical descriptions of BLDC motors, series-excited DC motors, and universal commutator motors, the proposed approach for synthesizing neural controllers can be applied to all of these machine types, as well as to other classes of controllers.

Using the developed approach for controller synthesis, we designed a control system for a DC drive of a complex mechanism equipped with a 3.5 MW motor with series excitation, which is subject to high accuracy requirements for following the specified tachogram (s-shaped diagram of start and braking to/from the nominal speed), including during rapid nominal load drops and surges in process of cutting rolling steel.

The DC drive model accounts for numerous characteristics of both the electrical and mechanical subsystems, including a three-loop cascade control structure for voltage, current, and speed regulation; the delay introduced by the thyristor voltage converter; the multi-mass kinematic configuration; and the dependence of the magnetic flux on the armature current.

As a result of combining a neural and fractional-integral speed controller, extremely efficient control performance is achieved. Figure 14 compares the transient response graphs and the speed error in the basic cascade control system (a) and in the system with the developed NN-PI-PI$^{\mu}$D speed controller (b).

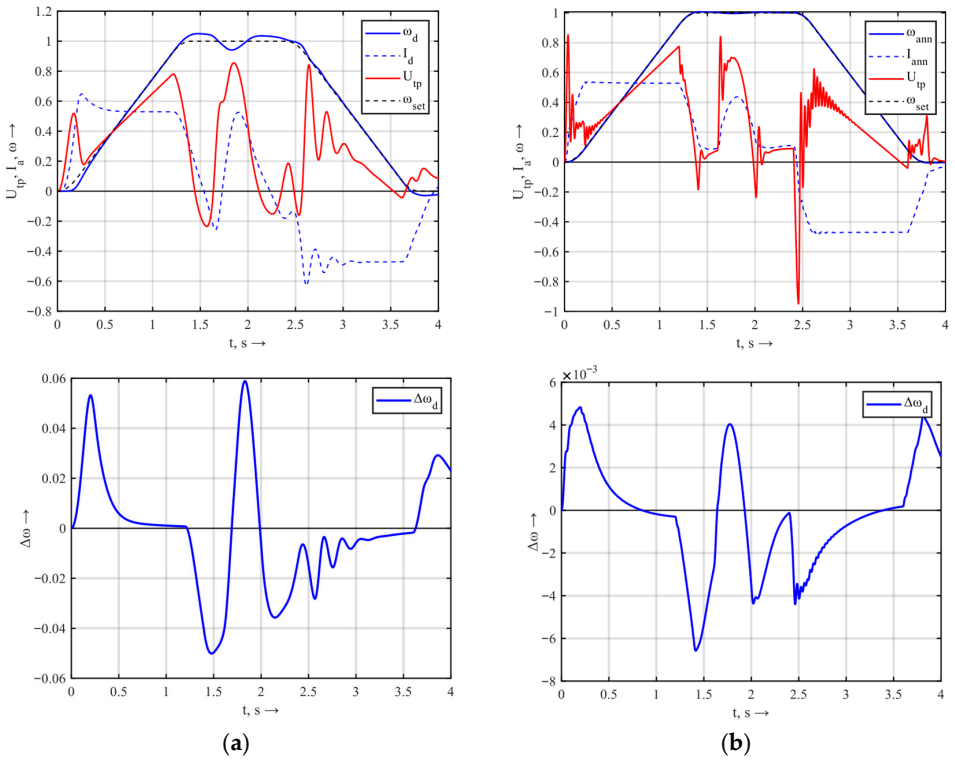


Figure 14. Transient response graphs in the basic cascade control system (**a**) and in the system with the NN-PI-PI$^{\mu}$D speed controller (**b**).

The speed error during start–stop operations decreased from 0.057 to 0.0048, and during load surges and drops from –0.048/+0.06 to –0.0067/+0.004 (in relative units of the nominal speed).

Verification of these statements, including experimental studies taking into account quantization in time and level in the control system, and the noise level of feedback signals, may be the subject of further research.

Author Contributions: Conceptualization, V.B. and V.K. (Valeriy Kuznetsov); methodology, V.B. and V.K. (Valeriy Kuznetsov); software, V.K. (Viktor Kovalenko), I.G. and V.K. (Valeriy Kuznetsov); validation, M.B. and V.K. (Viktor Kovalenko); formal analysis, V.B. and V.K. (Viktor Kovalenko); investigation, V.T. and V.K. (Valeriy Kuznetsov); resources, A.R., V.D. and Y.S.; data curation, V.D. and M.B.; writing—original draft preparation, V.K. (Viktor Kovalenko) and V.T.; writing—review and editing, V.T.; visualization, K.K. and Y.S.; supervision, A.R.; project administration, V.T. and M.B.; funding acquisition, K.K. and I.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: Author Valeriy Druzhinin was employed by the company Non-Profit Joint-Stock Company «Karaganda Industrial University». Author Yurii Shramko was employed by the company Technical University "Metinvest Polytechnic" LLC. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Appendix A

Sketch_fractional_integral.ino

```
#include "math.h"
double y [20];
double x [20];
double dx [20];
double xp [20];
double u_out;
double w_L = 0.01 * 0.01 * (sqrt(10.0));
double w_H = 100 * 100.0/(sqrt(10.0));
double ddL = 1;
int N = 8;
double mu = w_H/w_L;
double w_k, w_kp;
double r = −0.5;
double L0 = 0;
unsigned long time0, time1;
double ekp, ek;

void setup() {
 Serial.begin(115200);
 Serial.println(mu, 4);
 for (int k = -N; k <= N; k++) {
  ekp = ((k + N + 0.5 − 0.5 * r)/(2 * N + 1));
  ek = ((k + N + 0.5 + 0.5 * r)/(2 * N + 1));
  w_kp = pow(mu, ekp) * w_L;
```

```
 w_k = pow(mu, ek) ∗ w_L;
 L0 += log10(sqrt(1.0/w_kp/w_kp + 1.0)) − log10(sqrt(1.0/w_k/w_k + 1.0));
 }
 time0 = millis();
}

void loop() {
 double t;
 double dt = 1/w_kp/3.0;
 double Tmax = 0.96;

 double dT;

 int nnn = (0.02/dt);
 int i, j, k;

 x [1] = pow(10.0, (-L0));

 for (t = 0; t <= Tmax; t += nnn ∗ dt) {
  Serial.println(u_out, 4);
  for (int i = 1; i <= nnn; i++) {
   j = 1;
   for (k = N; k >= −N; k--) {
    ekp = ((k + N + 0.5 − 0.5 ∗ r)/(2 ∗ N + 1));
    ek = ((k + N + 0.5 + 0.5 ∗ r)/(2 ∗ N + 1));
    w_kp = pow(mu, ekp) ∗ w_L;
    w_k = pow(mu, ek) ∗ w_L;

    dx[j] = x[j] − xp[j];
    y[j] = 0.5 ∗ (y[j] ∗ (1 − dt ∗ w_k) + (dx[j]/w_kp + x[j] ∗ dt) ∗ w_k + (y[j] + (dx[j]/w_kp
+ x[j] ∗ dt) ∗ w_k)/(1 + w_k ∗ dt));
    x[j + 1] = y[j];
    xp[j] = x[j];
    j++;
   }
  }
  u_out = y[j − 1];
  time1 = millis();
  dT = (time1 − time0);
  time0 = time1;
 }
}
```

Sketch_ANN_learning.ino

```
#include “math.h”
double alfa = 0.2;
int N = 1; //input
int J = 5; //hidden
int K = 1; //output
int i, j, k, n;
```

```
double w2[10][1];
double w3[1][10];
double dE2[10][1];
double dE3[1][10];
double xk [1];
double xj [10];

double yset = −0.25;
double xset = +0.8;
double ya = 0;
double o3 = 0;
double er = 0;
double ej [10];
double o2[10];
int count = 0;
unsigned long time0;

void setup() {
 Serial.begin(115200);
 for (int j = 0; j < J; j++) {
  for (int k = 0; k < K; k++) {
   for (int n = 0; n < N; n++) {
    w2[j][n] = (double(rand())) ∗ 2.5/(double(RAND_MAX));
    w3[k][j] = (double(rand())) ∗ 2.5/(double(RAND_MAX));
   }
  }
 }
 time0 = millis();
}

void loop() {
//learning NN in Loops
 for (i = 0; i < 50; i++) {
  for (j = 0; j < J; j++) {
   o2[j] = 0;
   for (n = 0; n < N; n++) {
    o2[j] += w2[j][n] ∗ xset;
   }
   o2[j] = tanh(o2[j]);
  }

  o3 = 0;
  for (j = 0; j < J; j++) {
   for (k = 0; k < K; k++) {
    o3 += w3[k][j] ∗ o2[j];
   }
  }
  o3 = tanh(o3);
  ya = o3;
  if (count == 0) {
```

```
 Serial.println(yset − o3, 4);
}
er = yset − o3;

for (k = 0; k < K; k++) {
 xk[k] = 0;
 for (j = 0; j < J; j++) {
  xk[k] += w3[k][j] ∗ o2[j];
 }
 for (j = 0; j < J; j++) {
  ej[j] = 0;
  for (k = 0; k < K; k++) {
   ej[j] += er ∗ w3[k][j];
  }
  dE3[k][j] = -er ∗ (1 + tanh(xk[k])) ∗ (1 − tanh(xk[k])) ∗ o2[j];
  w3[k][j] += -alfa ∗ dE3[k][j];
 }
}

 for (j = 0; j < J; j++) {
  xj[j] = 0;
  for (n = 0; n < N; n++) {
   xj[j] += w2[j][n] ∗ xset;
  }
  for (n = 0; n < N; n++) {
   dE2[j][n] = -ej[j] ∗ (1 + tanh(xj[j])) ∗ (1 − tanh(xj[j])) ∗ xset;
   w2[j][n] += -alfa ∗ dE2[j][n];
  }
 }
}
if (count == 0) {
 unsigned long dT = millis() − time0;
 Serial.println(dT);
}
count = 1;
}
```

References

1. Lozynskyy, O.; Lozynskyy, A.; Kopchak, B.; Paranchuk, Y.; Kalenyuk, P.; Marushchak, Y. Synthesis and research of electromechanical systems described by fractional order transfer functions. In Proceedings of the 2017 International Conference on Modern Electrical and Energy Systems (MEES), Kremenchuk, Ukraine, 15–17 November 2017; pp. 16–19. [CrossRef]
2. Busher, V.; Zakharchenko, V.; Shestaka, A.; Kuznetsov, V.; Kuznetsov, V.; Nader, S. Optimization of the Control of Electromagnetic Brakes in the Stand for Tuning Internal Combustion Engines Using ID Regulators of Fractional Order. *Energies* **2022**, *15*, 9378. [CrossRef]
3. Busher, V.; Melnikova, L.; Horoshko, V. Synthesis and implementation of fractional-order controllers in a current circuit of the motor with series excitation. *East.-Eur. J. Enterp. Technol.* **2019**, *2*, 63–72. [CrossRef]
4. Chornyi, O.P.; Herasymenko, L.V.; Busher, V.V. The learning process simulation based on differential equations of fractional orders. *CTE Workshop Proc.* **2021**, *8*, 473–483. [CrossRef]

5. Podlubny, I. Fractional-order systems and PIλDμ-controllers. *IEEE Trans. Autom. Control* **1999**, *44*, 208–214. [CrossRef]
6. Oprzedkiewicz, K.; Mitkowski, W.; Gawin, E. Application of fractional order transfer functions to modeling of high—Order systems. In Proceedings of the 2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland, 24–27 August 2015; pp. 1169–1174. [CrossRef]
7. Caponetto, R.; Dongola, G.; Fortuna, L.; Petras, I. *Fractional Order Systems: Modeling and Control Applications*; World Scientific: Singapore, 2010; Volume 72.
8. Busher, V.; Aldairi, A. Synthesis and technical realization of control systems with discrete fractional integral-differentiating controllers. *East.-Eur. J. Enterp. Technol.* **2018**, *4*, 63–71. [CrossRef]
9. Xue, D.; Chen, Y.; Atherton, D.P. *Linear Feedback Control: Analysis and Design with MATLAB*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2008.
10. Chen, Y.Q.; Petráš, I.; Xue, D. Fractional order control—A tutorial. In Proceedings of the 2009 American Control Conference, St. Louis, MO, USA, 10–12 June 2009; pp. 1397–1411. [CrossRef]
11. Gude, J.J.; Camacho, O.; Di Teodoro, A.; García Bringas, P. Analytical method for the identification of higher-order fractional systems using fractional dual-pole plus dead-time models. *Results Eng.* **2025**, *25*, 105574. [CrossRef]
12. Ramanaiah, V.; Mandava, S. Performance analysis of permanent magnet synchronous motor based on transfer function model using PID controller tuned by Ziegler-Nichols method. *Results Eng.* **2025**, *26*, 105460. [CrossRef]
13. Bin Roslan, M.N.; Bingi, K.; Devan, P.A.M.; Ibrahim, R. Design and Development of Complex-Order PI-PD Controllers: Case Studies on Pressure and Flow Process Control. *Appl. Syst. Innov.* **2024**, *7*, 33. [CrossRef]
14. Nicola, M.; Nicola, C.-I.; Selișteanu, D.; Ionete, C.; Șendrescu, D. Improved Performance of the Permanent Magnet Synchronous Motor Sensorless Control System Based on Direct Torque Control Strategy and Sliding Mode Control Using Fractional Order and Fractal Dimension Calculus. *Appl. Sci.* **2024**, *14*, 8816. [CrossRef]
15. Nicola, M.; Nicola, C.-I.; Sacerdoțianu, D.; Vintilă, A. Comparative Performance of UPQC Control System Based on PI-GWO, Fractional Order Controllers, and Reinforcement Learning Agent. *Electronics* **2023**, *12*, 494. [CrossRef]
16. Eltayeb, A.; Ahmed, G.; Imran, I.H.; Alyazidi, N.M.; Abubaker, A. Comparative Analysis: Fractional PID vs. PID Controllers for Robotic Arm Using Genetic Algorithm Optimization. *Automation* **2024**, *5*, 230–245. [CrossRef]
17. Tytiuk, V.; Chornyi, O.; Baranovskaya, M.; Serhiienko, S.; Zachepa, I.; Tsvirkun, L.; Kuznetsov, V.; Tryputen, N. Synthesis of a fractional-order PIλDμ-controller for a closed system of switched reluctance motor control. *East.-Eur. J. Enterp. Technol.* **2019**, *2*, 35–42. [CrossRef]
18. Mohamed, M.J.; Oleiwi, B.K.; Abood, L.H.; Azar, A.T.; Hameed, I.A. Neural Fractional Order PID Controllers Design for 2-Link Rigid Robot Manipulator. *Fractal Fract.* **2023**, *7*, 693. [CrossRef]
19. Oustaloup, A.; Melchior, P.; Lanusse, P.; Cois, O.; Dancla, F. The CRONE toolbox for Matlab. In Proceedings of the IEEE International Symposium on Computer Aided Control System Design (CACSD 2000), Anchorage, AK, USA, 25–27 September 2000; pp. 190–195.
20. Valério, D. Toolbox Ninteger for MatLab (Version 2.3) [Computer Software]. 2005. Available online: http://web.ist.utl.pt/duarte.valerio/ninteger/ninteger.htm (accessed on 19 November 2025).
21. Oldenhuis, R. Optimize [Computer Software]. MathWorks File Exchange. 2009. Available online: https://www.mathworks.com/matlabcentral/fileexchange/180212-optimization (accessed on 19 November 2025).
22. Gu, D.; Hu, H. Neural predictive control for a car-like mobile robot. *Robot. Auton. Syst.* **2002**, *39*, 73–86. [CrossRef]
23. Morgado-Dias, F.; Mota, A. Comparison Between Different Control Strategies Using Neural Networks. 2001. Available online: https://www.researchgate.net/publication/310047342_Comparison_between_different_Control_Strategies_using_Neural_Networks (accessed on 19 November 2025).
24. Venayagamoorthy, G.K.; Harley, R.G.; Wunsch, D.C. Implementation of adaptive critic-based neurocontrollers for turbogenerators in a multimachine power system. *IEEE Trans. Neural Netw.* **2003**, *14*, 1047–1064. [CrossRef] [PubMed]
25. Goering, S.; Klein, E.; Sullivan, L.S.; Wexler, A.; Arcas, B.A.Y.; Bi, G.; Carmena, J.M.; Fins, J.J.; Friesen, P.; Gallant, J.; et al. Recommendations for Responsible Development and Application of Neurotechnologies. *Neuroethics* **2021**, *14*, 365–386. [CrossRef] [PubMed]
26. Si, J.; Barto, A.G.; Powell, W.B.; Wunsch, D.C. Model-Based Adaptive Critic Designs. In *Handbook of Learning and Approximate Dynamic Programming*; IEEE Press: Piscataway, NJ, USA, 2004; pp. 65–95. [CrossRef]
27. Akhyar, S.; Omatu, S. Neuromorphic self-tuning PID controller. In Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, USA, 28 March–1 April 1993; pp. 1076–1080. [CrossRef]

28. D'Emilia, G.; Marra, A.; Natale, E. Use of neural networks for quick and accurate auto tuning of PID controller. *Robot. Comput.-Integr. Manuf.* **2007**, *23*, 170–179. [CrossRef]
29. Rashid, T. *Make Your Own Neural Network*; CreateSpace Independent Publishing Platform: Scotts Valley, CA, USA, 2016; 222p, ISBN 10: 1530826608/13: 978 1530826605.