Adaptive neural network based control of balancing robot in real time mode

A I Glushchenko[1], V A Petrov[1] and K A Lastochkin[1]

[1] Automated and information control systems department, Stary Oskol technological institute n.a. A.A Ugarov (branch) NUST "MISIS", Stary Oskol, Russia

Abstract. The scope of this research is to control a balancing robot in real time mode. This problem is solved with the help of a neural network, which is trained online and used as a controller. A method to develop such controller is proposed. Particularly, the neural network structure is chosen, restrictions is developed to determine situations when to train the network online, an algorithm is proposed to define the sign of change in the network weights. The obtained controller is compared to a linearly quadratic regulator (LQR) using a real balancing robot on LEGO EV3 platform. Experiments are conducted in two modes. The first of them is to keep the robot at an unstable equilibrium mode. The second one is both to make the robot follow the user's setpoint for the state coordinates and stabilize the plant. The obtained results show that the control quality is improved comparing to the LQR controller, since the system with the neural network is able to adapt to real conditions of the experiments. As far as the first type of experiments is concerned, the robot controlled by the network has covered the distance, which is by 1630 radians shorter comparing to the LQR controller.

1. Introduction

In this research, the problem of a balancing two-wheeled robot control is solved. This robot is a physical implementation of an inverted pendulum on a trolley. Various methods to control such plant are developed [1-3]. However, this task has not been fully solved by now. At the same time, the efficiency improvement of the balancing robots control would allow to use them to create vehicles for people with disabilities, robotic loaders for storage facilities and planetary rovers for the space industry [4].

In general, the existing control algorithms of the considered plant can be divided into two groups.

Classic control methods include PID [5], LQR controllers [6] and H∞ theory [7]. The main problem of their implementation for the real balancing robots is that the parameters of the above mentioned controllers are calculated using linearized plant models and not adjusted during operation. This may result in the control system instability due to the existence of the plant nonlinearities and non-stationarity, and changes in the environment conditions.

To some extent, the above mentioned problem could be overcome with the help of the intelligent control methods. Among such methods, neural networks and fuzzy logic are commonly applied for the balancing robot control. They have the ability for online training and / or allow to take knowledge about the plant into consideration. They also have adaptive properties. Neural networks in a control loop are used both as regulators and, in order to adjust the existing controllers, neural network tuners.

For example, a neural network adjuster of the PID controller for the balancing robot is developed in [8]. In [9, 10] a method of neural network controller synthesis is proposed. The disadvantages of these techniques are as follows. As for the systems using off-line training, it is the complexity of obtaining

of a training set (either an accurate model of the plant is needed, or the samples are formed with the help of an existing controller, and this will not allow to create a more accurate regulator). As for the systems using online training, it is the absence of restrictions on both the learning rate for the neural network and time moments when to train it. Such systems do not also take into account the features of the control objects in question (a priori knowledge about them). For these reasons, the considered control systems can be used for the models of the inverted pendulums, but their application to the real control object faces certain difficulties. Another promising intelligent method is a fuzzy logic, which is often implemented as fuzzy controllers [11-13]. The disadvantage of the fuzzy logic is that the values of the normalization coefficients of the input and output values of the fuzzy controller are to be experimentally found. It is also possible that these coefficients will have to be adjusted during plant functioning.

Thus, only a small part of the above mentioned algorithms, both classical and intelligent, can be implemented for a real control object – the balancing robot. But, in general, the neural networks application seems to be the most promising to solve the considered problem because of their ability to approximate dependencies and be trained online.

In this research, it is proposed to combine a neural network, functioning as a controller, with a base of conditions and restrictions on the online training of such network. This combination of intelligent approaches will provide the controller with the adaptive properties, and it will not have the above mentioned disadvantages.

2. Control object description

A real balancing robot based on the LEGO EV3 platform has been selected as a control object. Its overall view is shown in Figure 1, its kinematic scheme (a side view) – in Figure 2, its top view – in Figure 3.

Figure 1. Robot overall view.

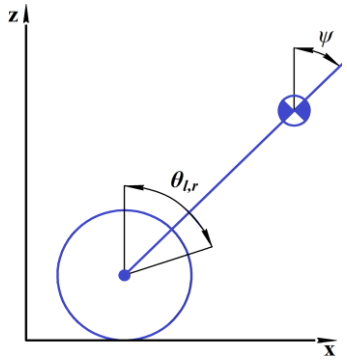

Figure 2. Robot kinematic scheme (side view).

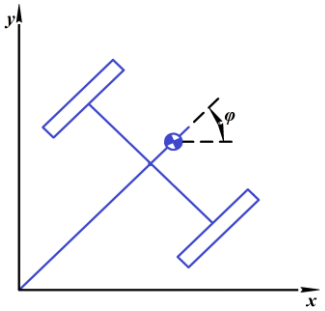

Figure 3. Robot top view.

Considered control object is described by seven coordinates in the state space. θ (*theta*) is the mean angle of wheels turning (an average value between the angular positions θ_l and θ_r of the left and right wheels) $\theta = 0.5 \cdot (\theta_l + \theta_r)$, θ' (*theta_dot*) is the speed of the wheels rotation, θ_{int} (*theta_int*) is the integral of θ, ψ (*psi*) is the body pitch angle, ψ' (*psi_dot*) is the speed of ψ value change, φ (*phi*) is the robot yaw angle, φ'(*phi_dot*) is the speed of φ value change. The task is to control all seven state

coordinates at the same time. In this case, the control action for the robot, which has only two actuators, is the vector of voltage for the left and right motors ($\boldsymbol{u} = (u_l, u_r)$).

3. Neural network controller synthesis

The first step of the neural network controller synthesis is the choice of the network structure, as well as the selection of input and output values.

As it is mentioned above, the balancing robot is a multi-loop control object. So it requires simultaneous control of all seven state coordinates for normal operation. As far as the classic theory of automatic control is concerned, the controller for such objects is developed in the form of a certain matrix of coefficients for all components of the control system negative feedback. Such matrix is called a linearly quadratic regulator (LQR) [14]. This controller obtains the vector consisting of the control errors of all state coordinates $\boldsymbol{E} = [e_{\theta int}\ e_\theta\ e_\psi\ e_{\theta'}\ e_{\psi'}\ e_\varphi\ e_{\varphi'}]^T$. Its output is a control action vector $\boldsymbol{u} = (u_l, u_r)$. The parameters matrix of the LQR controller, which components remain constants during the whole period of the plant functioning, is calculated on the basis of the control object state space model [15] by minimization of the squared optimality criterion (1).

$$J = 0.5 \cdot \int_0^\infty (E^T Q E + u^T R u) dt \to \min, \tag{1}$$

where $\boldsymbol{Q}(7,7)$ and $\boldsymbol{R}(2,2)$ are positively defined identity matrixes.

The structure of the neural network for the proposed controller is chosen in such a way that the mathematical operations in its output layer repeat such operations of the LQR controller. The control error vector $\boldsymbol{E}$ containing seven elements is sent to the input of the neural network, because the neural controller is to have no less information than the considered optimal controller has. This optimal controller is used as a basis to develop nonlinear adaptive neural network controller. So the number of input neurons is chosen to be seven. The hidden layer also requires seven neurons to transmit the vector $\boldsymbol{E}$ to the output layer. The number of output neurons corresponds to the number of the control actions used in the control system under consideration. It equals to two. The hidden and output layers have linear activation functions to repeat the mathematical operations of the LQR controller.

The next step after choosing the structure of the neural network is to train it. There are different techniques to implement that [16]. And, first of all, they can be divided into methods of offline and online training. The aim of the first of them is to ensure the correct functioning of the network at the moment of its integration into the balancing robot control system. The best course of action in this case is to use the parameters values found for the classical controller – LQR. The network weights are artificially set in such a way that the elements of the weight matrix of the output layer $\boldsymbol{LW}(7,2)$ coincide with the corresponding elements of the parameters matrix of the LQR controller. The weight matrix of the hidden layer $\boldsymbol{IW}(7,7)$ is an identity matrix. Considering that the linear activation functions are used in the hidden and output layers, the biases are equaled to zero.

Thus, the calculated matrix of weights of the output layer $\boldsymbol{LW}$ is shown as (2).

$$LW = \begin{bmatrix} 0.644 & 1.242 & 59.38 & 1.391 & 7.1 & 0.677 & 0.179 \\ 0.644 & 1.242 & 59.38 & 1.391 & 7.1 & -0.677 & -0.179 \end{bmatrix}. \tag{2}$$

The neural network of the selected structure is depicted in Figure 4.

Taking the non-stationarity of the control object into consideration, the next step of the controller synthesis is to arrange the online training of the neural network.

4. Online neural network training

Online training of the chosen neural network is performed according to the backpropagation algorithm as one of the most widely used methods [16]. At the same time, the training error (according to which the weights are adjusted) is calculated as follows. The error is ½ of the sum of the squared distances between the current control object state coordinates value and the required ones – equation (3).

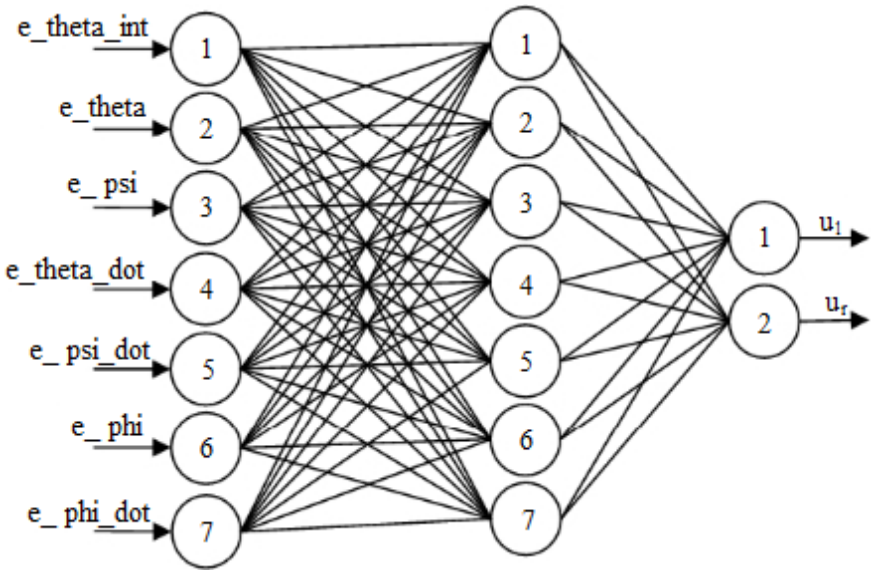

Figure 4. Neural network controller structure.

It is proposed to calculate it using the coordinates θ', ψ', φ', since their values are directly obtained from the sensors of the real robot (a gyroscope and two encoders).

$$E_T(t) = 0.5 \cdot (e^2_{\theta'}(t) + e^2_{\psi'}(t) + e^2_{\varphi'}(t)) \tag{3}$$

Considering the real control object, the error value (3) is not possible to be reduced to zero due to the measurement error and physical features of the robot. So the permissible level N_e of the training error (3) should be selected. It has been equaled to eight for the experiments in this research due to the amplitude of the sensors noise. Thus, the basic rule used to execute the online training is formulated as follows: the neural network controller is trained if the total error $\boldsymbol{E}_\mathbf{T}(t)$ is greater than N_e units. Learning rates for the output and hidden layers are experimentally equaled to $\eta_{LW} = 10^{-7}$; $\eta_{IW} = 10^{-7}$ respectively. These values are not going to be corrected during experiments.

The process of functioning of the balancing robot can be divided broadly into two main modes: 1) stabilization mode, when the control system is required to ensure the stability of the control object under the condition that the setpoint values for all state coordinates are equaled to nil; 2) the mode when the user defines the setpoint values of the coordinates θ' and/or φ'. Each mode needs its own "best" set of the neural network weights (controller parameters) values. The control error of some coordinates can influence the control quality in different ways for different modes. So the algorithm to select the sign of change in neural network output layer weights is developed and shown in Figure 5. Here $\boldsymbol{E}_\mathbf{T}(t)$ is the training error (3), *Ref* θ' is the setpoint value of θ' state coordinate, $\Delta\omega_{\theta int}$ is the change in output layer weight, which is responsible for θ_{int} coordinate, $\Delta\omega_\theta$ is the change in value of the output layer weight, which is responsible for θ coordinate, $\Delta\omega_{\theta'}$ is the change in value of the output layer weight, which is responsible for θ' coordinate.

This algorithm allows to adjust the current weight value so as to reduce the error (3) and provide the required quality of control, taking into account the current functioning mode of the control object.

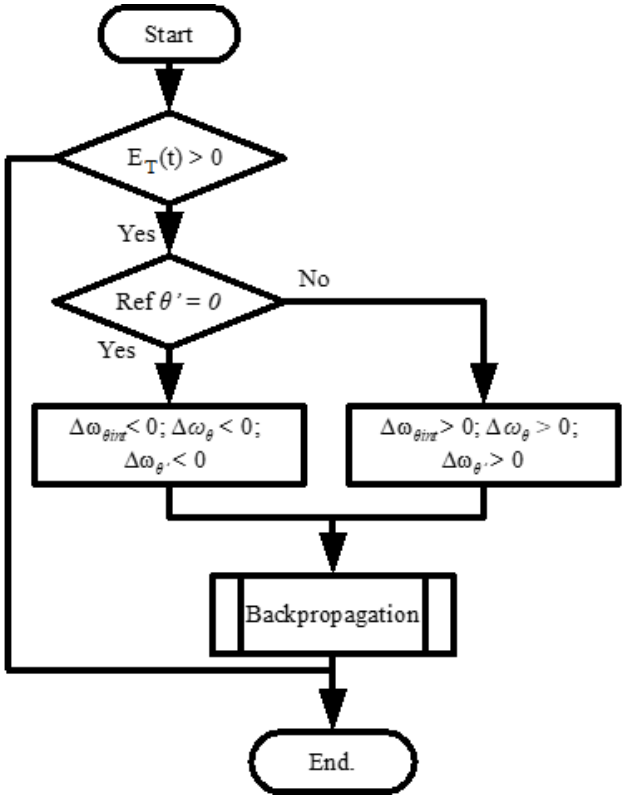

Figure 5. Algorithm of sign selection of change in value of output layer weights.

The algorithm shown in Figure 5 is based on data of LQR controller functioning obtained experimentally and a priori known data of the control object. A posteriori, it has been found that if the current robot functioning mode is to follow the user's setpoint, and the output layer weight for the control error of the coordinate θ_{int} is going to zero, then the control quality will be improved. But if the stabilization mode is current, then the same actions will have vice versa effect on the control quality. It is also a priori known that the signs of changes in output layer weights, which are responsible for θ, θ_{int} and θ' coordinates, must be of the same sign.

5. Experiments with real balancing robot

The robot control systems have been developed in the MatLab Simulink software. The connection of the LEGO EV3 microprocessor to the MatLab Simulink to conduct experiments with the real control object was established using an Ethernet cable. Discretization interval (T_s) of EV3 was 0.004 s. The same time interval was set for calculations during the simulations in Matlab.

5.1. Stabilization mode experiments

The neural network controller has been required to provide better control quality in comparison with the LQR controller. The setpoint values for all the state coordinates were equaled to nil. The duration of the experiment was 195 seconds. Obtained results for six of seven state coordinates for LQR is shown in Figure 6, a neural network controller – in Figure 7. Transients curves for the coordinate φ' for both controllers can be described as constant oscillations of one radian/s amplitude with mean value of zero radian/s and are not shown in Figure 6 and Figure 7.

Evaluation of the proposed neural network controller effectiveness was made by comparison of the final values of the coordinates θ_{int} for both considered controllers. This coordinate showed the total distance covered by the robot during the experiment. As this is the stabilization mode, then the shorter the distance, the better.

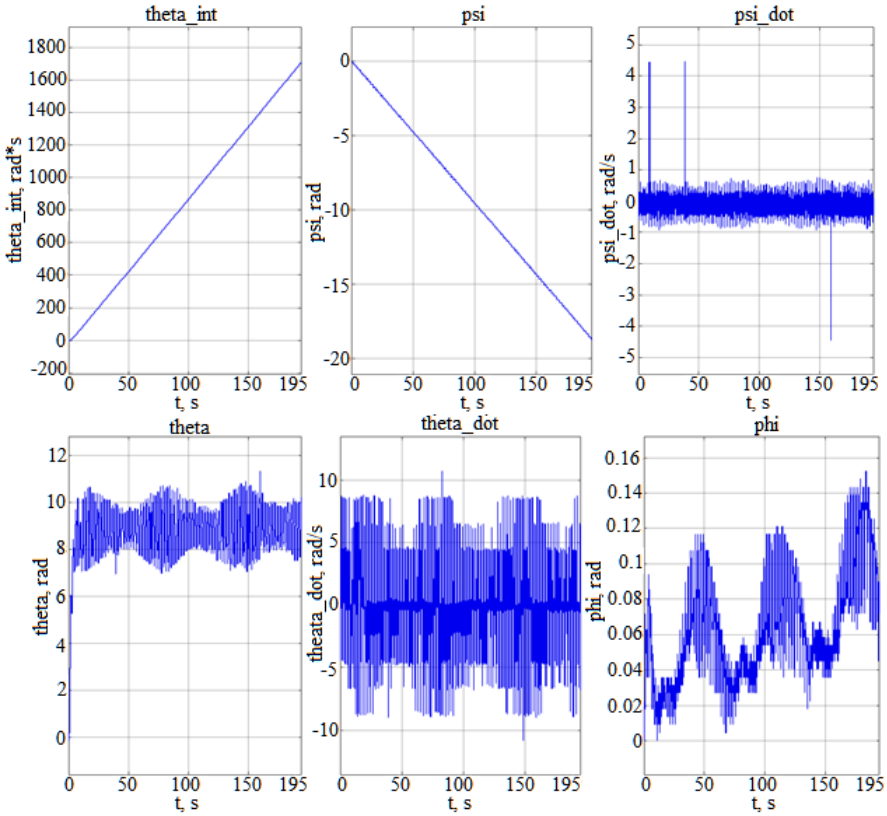

Figure 6. Transients of state coordinates obtained with LQR controller (stabilization mode).

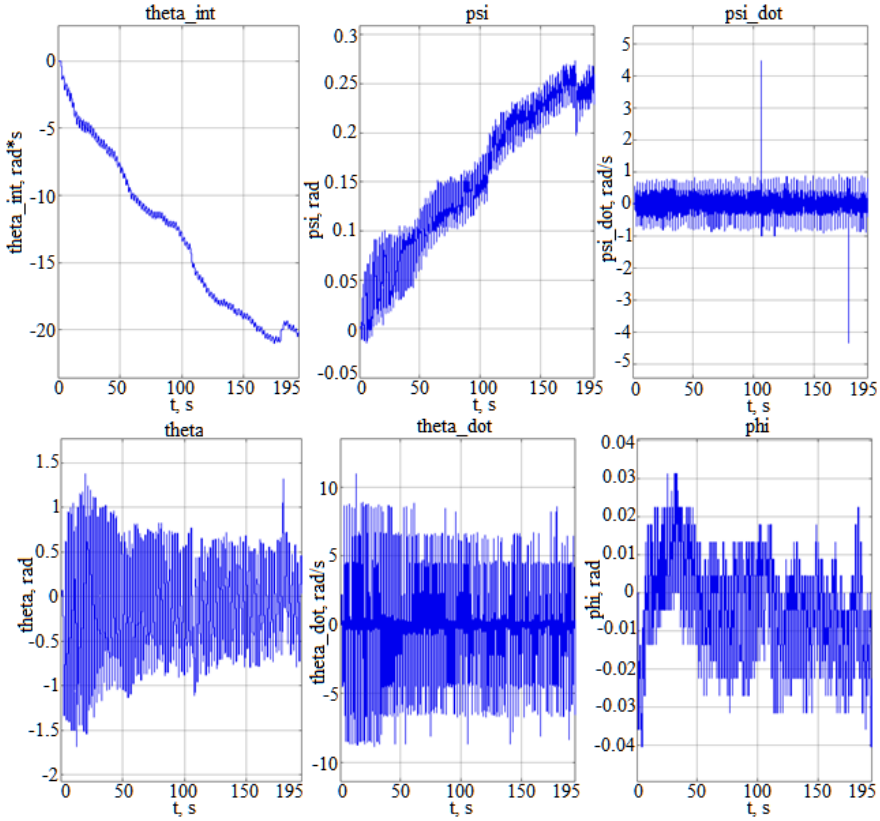

Figure 7. Transients of state coordinates obtained with neural network controller (stabilization mode).

Having analyzed θ_{int} curves in Figure 6 and Figure 7, the conclusion was made that the robot controlled by the neural network controller covered a shorter distance (20 radians in 195 seconds) than the one with the LQR controller (1650 radians in 195 seconds). There is also a noticeable decrease in the absolute value of the final value of ψ coordinate. This increases the stability of the control object. Visual observations during the experiments with both controllers confirmed the data obtained from the sensors. Figure 8 shows a graph of the training error change throughout the experiment, and Figure 9 shows the change in the weight coefficients for neurons, which are responsible for the coordinates θ, θ_{int}, θ'.

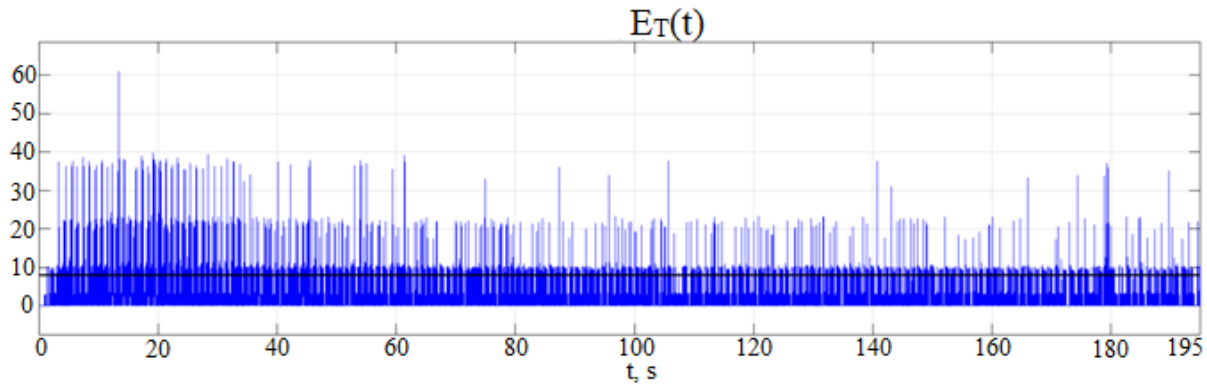

Figure 8. $\boldsymbol{E_T}(t)$ curve for experiment with neural network controller.

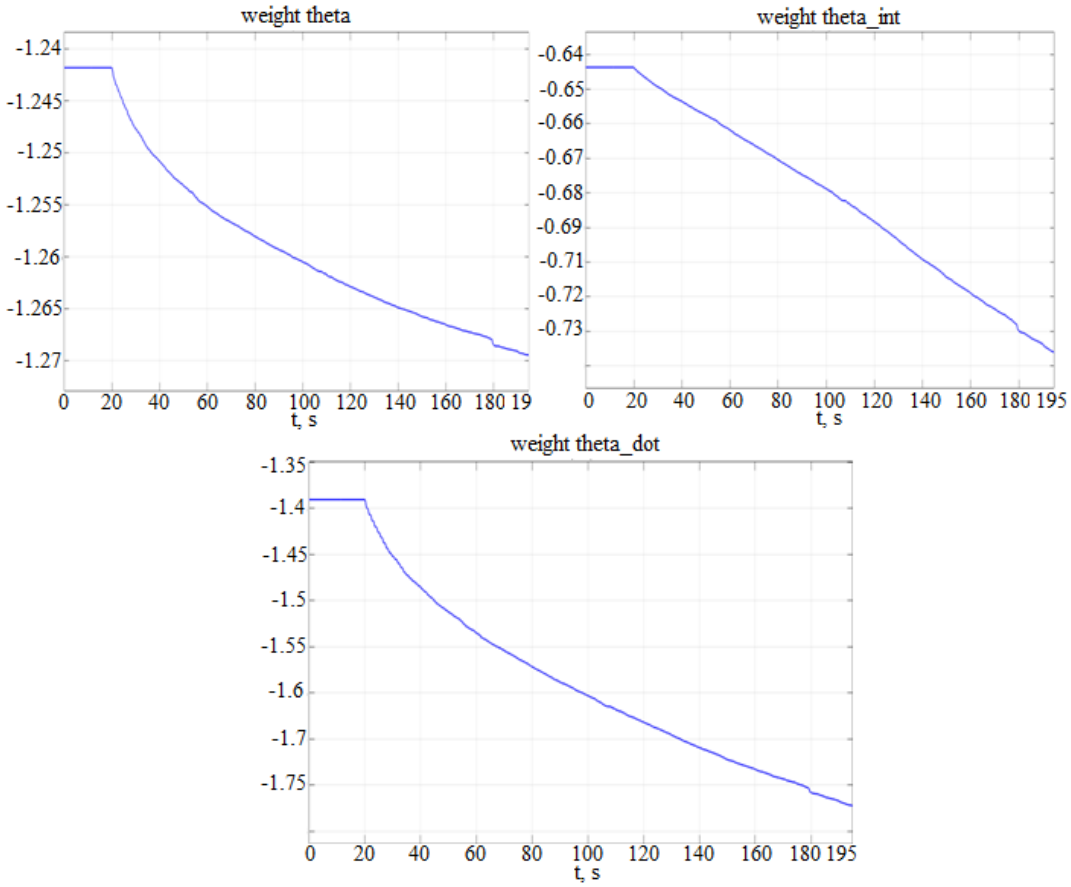

Figure 9. Change in output layer weights, which are responsible for θ, θ_{int}, θ' coordinates (stabilization mode).

The graph in Figure 8 shows a decrease of $\boldsymbol{E}_{\mathbf{T}}(t)$ value after 35 seconds of the experiment. It is not depicted in Figure 8, but $\boldsymbol{E}_{\mathbf{T}}(t)$ curve for the LQR controller did not change its amplitude and kept the same form during the whole experiment as in Figure 8 before the 35th second. The graph in Figure 9 illustrates the results of the online training according to the developed algorithm (Figure 5).

Thus, experimental results confirm the effectiveness of the proposed neural network controller and the algorithm to choose the sign of the changes in weights for the stabilization mode.

5.2. Experiment combining stabilization and setpoint following modes – mixed mode

This mode of the balancing robot functioning combines the two main modes described earlier. This means that at some moments the robot operates in the stabilization mode, while at other moments – in the mode of the user's setpoint following. It is very common for real two-wheeled vehicles. The aim of this experiment was to test the ability of the neural network controller to reconfigure the weights according to the developed algorithm in Figure 5. In this case, the comparison of transient processes quality obtained with the help of the neural network and LQR controllers was not performed due to the complexity of accurate reproduction of the experiment. Figure 10 shows the transients for the control system with the neural network controller. The graph with θ (*theta*) curve is divided into sections, each of which corresponds to a certain mode – stabilization one (number "0") and setpoint following one (number "1"). During the experiment, the θ' setpoint value was changed from zero (stabilization mode) to one (setpoint following mode) and back.

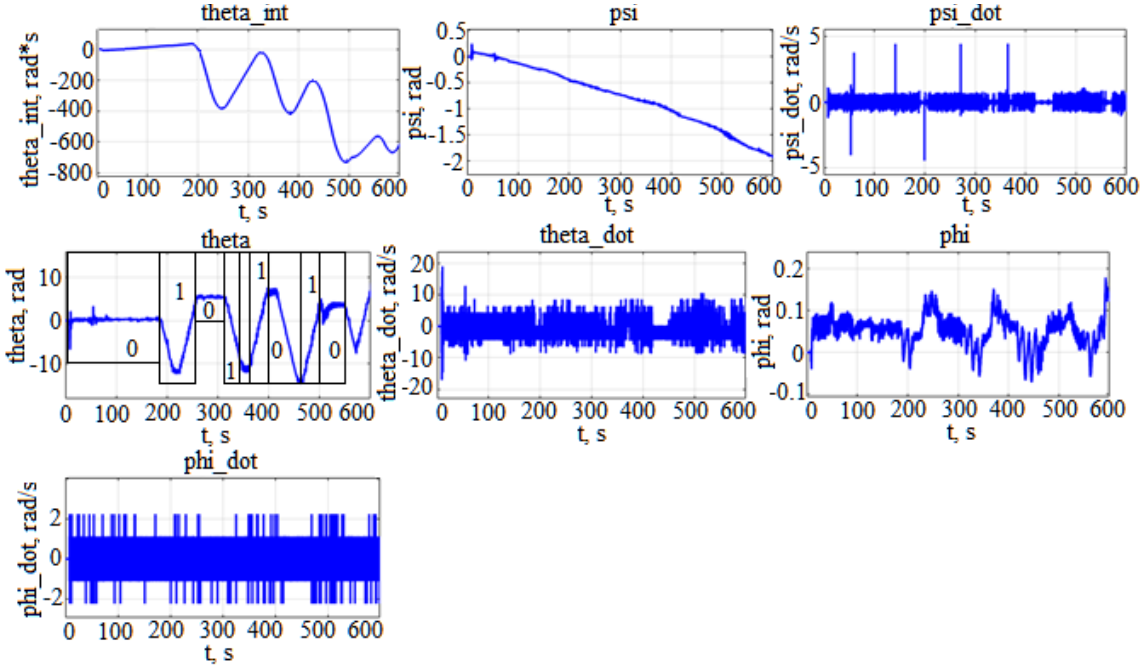

Figure 10. Neural network controller results obtained in mixed mode.

Changes in the weights of the neurons, which were responsible for the coordinates θ, θ_{int}, θ', in mixed mode with a division into corresponding sections for each particular mode (number "0" is the stabilization mode; number "1" – the setpoint following mode) are shown in Figure 11 and Figure 12.

It can be concluded from the curves in these figures that: 1) the weights decreased in the stabilization mode, 2) the weights increased in the setpoint following mode. That corresponds to the logic of the developed algorithm depicted in Figure 3.

Figure 13 shows the training error $\boldsymbol{E}_{\mathbf{T}}(t)$ curve with the division into the above described sections. For each mode section the final value of $\boldsymbol{E}_{\mathbf{T}}(t)$ became close to the required value of 8 units as a result of neural network training. The smallest value of error was achieved at time moments of 420-470 seconds. It can be concluded from Figure 10 that the best quality of control of the balancing robot is

achieved at these moments of time. This resulted in a minimum value of amplitude oscillations of θ' and ψ' coordinates.

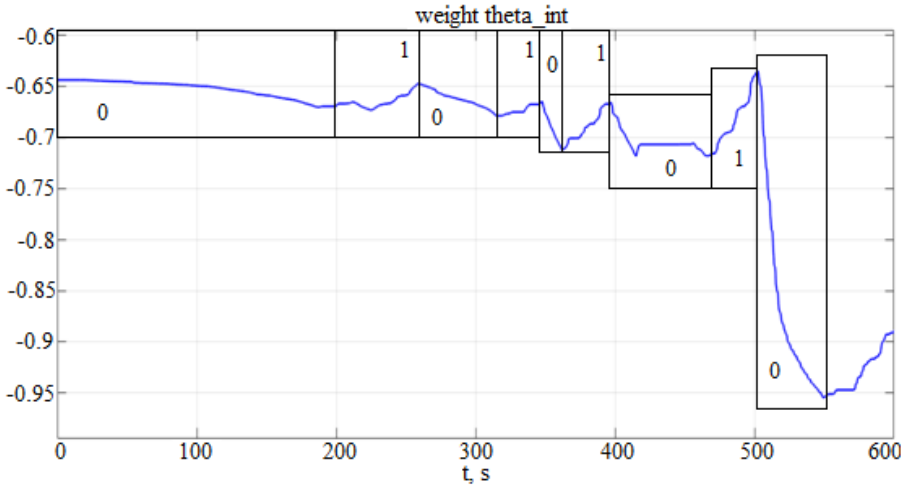

Figure 11. Change in weigh, which is responsible for θ_{int}, in mixed mode.

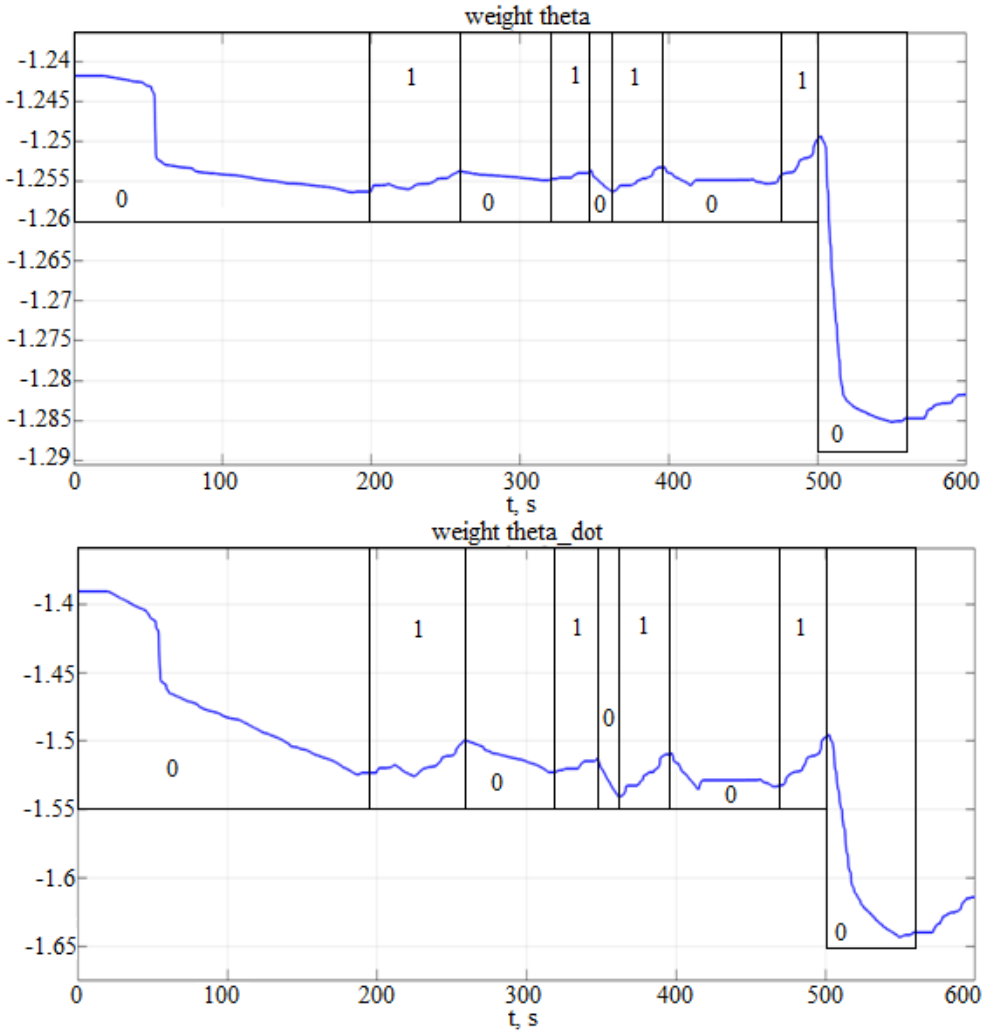

Figure 12. Change in weighs, which are responsible for θ and θ', in mixed mode.

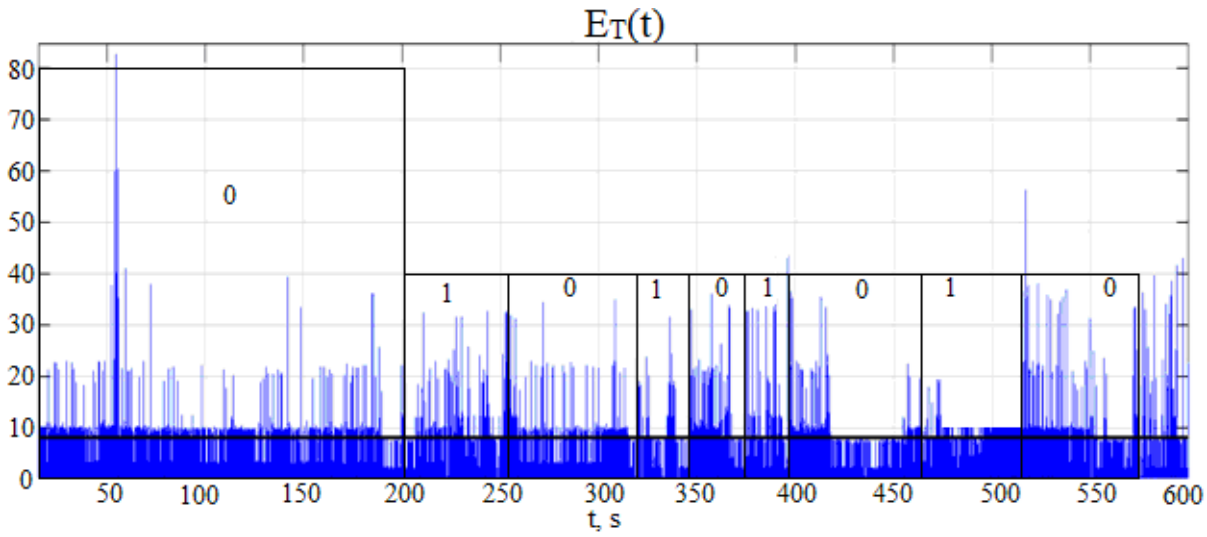

Figure 13. Training error $\boldsymbol{E}_{\mathrm{T}}(t)$ in mixed mode.

Thus, the results of the experiment in the mixed mode demonstrate the change in the output layer weights in accordance with the proposed algorithm and the reduction of the training error $\boldsymbol{E}_{\mathrm{T}}(t)$ making it close to the required value. These mean that the proposed neural network controller is effective to control the balancing robot functioning in different modes.

6. Conclusion

In this research, the neural network controller synthesis method to control the balancing robot in real time was developed. The set of restrictions on the online training of such regulator was proposed, the algorithm to choose the sign of change in output layer weights was devised. The obtained method allowed to improve the quality of control of the robot both for the stabilization and setpoint following robot functioning modes.

As for the stabilization mode, the experimental results showed the effectiveness of the developed controller as the control system with the neural network controller covered shorter distance (20 radian in 195 seconds) comparing to the LQR controller (1650 radian in 195 seconds).

The scope of further research is to expand the base of restrictions on online network training through: 1) the development of stability assessment criterion for the system with the neural network controller of the balancing robot on the basis of the second Lyapunov's method, 2) modification of the algorithm to choose the sign of change in the output layer weights.

7. References

[1] Semenov M E, Solovyov A M and Meleshenko P A 2015 Elastic inverted pendulum with backlash in suspension: stabilization problem *Nonlinear Dynamics* **82** pp 677–688.

[2] White W and Fales R 1999 Control of double inverted pendulum with hydraulic actuation: a case study *Proc. American Control Conference (San Diego)* (IEEE) pp 495–499.

[3] Spong M W 1995 The swing up control problem for the acrobat *IEEE Control Systems Magazine* **15** pp 72–79.

[4] Chan R P M., Stol K A and Halkyard C R 2013 Review of modelling and control of two-wheeled robots *Annual Reviews in Control* **37** 1 pp 89-103

[5] Sung H C 2015 *Balancing Robot Control and Implementation. Master's thesis* (Texas: A & M University).

[6] Sun L and Gan J 2010 Researching of two-wheeled self-balancing robot base on LQR combined with PID. *Proc. 2nd International Workshop on Intelligent Systems and Applications (Wuhan)* (IEEE) pp 1-5.

[7] Ruan X and Chen J 2010 H∞ robust control of self-balancing two-wheeled robot *Proc. 8th World Congress on Intelligent Control and Automation (Jinan)* (IEEE) pp 6524-6527.
[8] Ren T J, Chen T C and Chen C J 2008 Motion control for a two-wheeled vehicle using a self-tuning PID controller *Control Engineering Practice* **16** 3 pp 365-375.
[9] Noh J S, Lee G H and Jung S 2010 Position control of a mobile inverted pendulum system using radial basis function network *International Journal of Control, Automation and Systems* **8** 1 pp 157-162.
[10] Jung S and Kim S S 2008 Control experiment of a wheel-driven mobile inverted pendulum using neural network *IEEE Transactions on Control Systems Technology* **16** 2 pp 297-303.
[11] Nasir A N K *et al.* 2011 Performance comparison between fuzzy logic controller (FLC) and PID controller for a highly nonlinear two-wheels balancing robot *Proc. First International Conference on Informatics and Computational Intelligence (Bandung)* (IEEE) pp 176-181.
[12] Wu J, Zhang W 2011 Design of fuzzy logic controller for two-wheeled self-balancing robot *6th International Forum on Strategic Technology* (*Harbin*) vol 2 (IEEE) pp 1266-1270.
[13] Azizan H *et al.* 2010 Fuzzy control based on LMI approach and fuzzy interpretation of the rider input for two wheeled balancing human transporter 2010 *Proc. 8th IEEE International Conference on Control and Automation (Xiamen)* (IEEE) pp 192-197.
[14] Zhou K, Doyle J C and Glover K 1996 *Robust and optimal control* (New Jersey: Prentice hall).
[15] Yamamoto Y 2008 *NXTway-GS Model-Based Design-Control of self-balancing two-wheeled robot built with LEGO Mindstorms NXT* (Cybernet Systems Co., Ltd).
[16] Reed R and Marks II R J 1999 *Neural smithing: supervised learning in feedforward artificial neural networks* (MIT Press).

Acknowledgments
This work was supported by the Russian Foundation for Basic Research. Grant No 18-47-310003.