# The simulator and neuro-controller for small satellite attitude development

Nataliya Shakhovska[0000-0002-6875-8534], Dmytro Kozii, Pavlo Mukalov

Lviv Polytechnic National University, Lviv 79013, Ukraine
nataliya.b.shakhovska@lpnu.ua, dmytruto@gmail.com,
pmykalov@gmail.com

**Abstract.** The paper describes the realization of simulator and neuro-controller for small satellite attitude. The main types of neuro-controllers are analyzed. A problem of proper neuroemulator choosing for neurocontroller training is analyzed. A new criterion on the basis of local control gradients analysis for input neuroemulator's neurons is proposed. Results of numerical simulations of neurocontroller training by a gradient descent method is given.

**Keywords:** satellite, neuro-controller, learning rate, attitude

## 1 Introduction

Neural control is a kind of adaptive control when artificial neural networks (NN) are used as building blocks of control systems. Neural networks have a number of unique properties that make them a powerful tool for building control systems: the ability to learn from examples and to summarize data, the ability to adapt to changing properties of the object of control and the environment, the suitability for the synthesis of nonlinear regulators. Over the past 20 years, a large number of neurological methods have been developed, the most popular among them are Model Reference Adaptive Neurocontrol and Adaptive Critics [2].

The method of neural control with a reference model, also known as a "circuit with neurotransmitter and neuro-controller" or "reciprocal distribution in time," was proposed in the early 1990s [1], [3 – 5]. This method does not require knowledge of the mathematical model of the control object. Instead, a separate neural network, a neuromuscular, studies the direct dynamics of the control object and then it is used to calculate derivatives when training a neuro-controller. At the same time, the trained neuro-emulators with the lowest mean square error of the simulation of the control object usually chooses from the set of trained neuro-emulators. However, is this criterion best if the neural network is used for further training another neural network, connected sequentially to the first, and not actually for modeling the control object?

The paper presents neurocontroller development for satellite rotation control.

## 2 State of arts

NN was proposed in 1943. McCullock and Pitts as the result of studying the structure and activity of biological neurons.

A typical structure of the automatic control system with the PID-regulator and the NN as an automatic adjustment unit is considered in the work [6]. NN acts as a functional transformation, which for each set of input signals the coefficients for the PID regulator are produced. The most complicated part of the design of an NN-based regulator is the training procedure, which reduces to the identification of unknown NN parameters, such as weighting factors and displacement of neurons. For NN training, the gradient search method uses a minimum criterion function, which depends on the parameters of the neurons. The search process is inertial, at each iteration, the search for all coefficients of the network occurs: first for the output layer, then for the previous and so on to the first.

The length of the learning process is a key issue when using NN methods for PID regulators [7]. In addition, when applying NN, there are difficulties due to the impossibility of predicting regulation errors for incoming actions that were not included in the set of training sequences by determining the structure of neurons in the network, the duration of training, the range and the number of training actions.

The main purpose of NM training is to choose the weighting factors of such a network to ensure consistency between input and output values. The neuron with the input $p = \{p_1, p_2, ..., p_r\}$ is shown in Fig. 1. The initial value is equal to the scalar product of the vector $W$ on the input vector $p$, the bias value $b$ is added to the weighted sum of inputs [8].

Output signal is:

$$n = w_{11} \cdot p_1 + w_{12} \cdot p_2 + ... + w_{1R} \cdot p_R + b \quad (1)$$

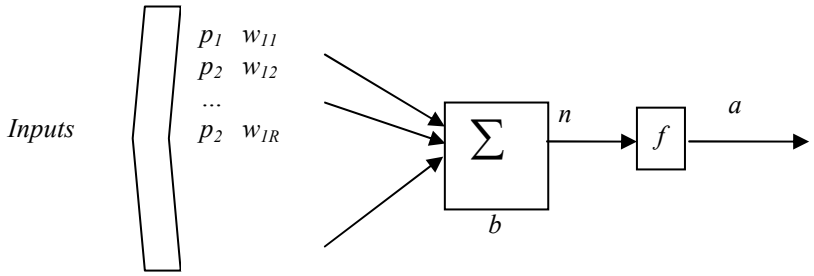


**Fig. 1.** Neuron structure

The choice of NN architecture is to determine the number of layers, the number of neurons in each of the layers, the form of the activation function of each layer, and information about the topological links of the neurons. Single-layer NNs are not suitable for solving complex problems [9], but combining several neurons into one or more layers has great potential. The two-layer NN, which in the first layer contains a

sigmoidal activation function, and in the second one linear, can be trained to approximate any function with finite number of breakpoints with arbitrary accuracy [9].

The purpose of identification is to determine the operator of the model, which converts the input action of the controlled object to the output value. Different identification methods are possible depending on the various forms of representation of mathematical models in the form of ordinary differential equations, difference equations, convolution equations [10], and others. However, none of the proposed methods is universal.

The paper [11] considers the use of NN as an alternative tool for the identification of dynamic objects. The use of NN is based on the fact that in practice modern electric drives are multi-mass systems with nonlinear links. Relevant linearized models built based on transfer functions, cannot always adequately reflect the state of the electric drive in all modes of its operation. The equivalence of a nonlinear system and its linear approximation will be equal in a limited time interval, and when transitioning the output system from one mode to another, it is expedient to use the linearization method and obtain a new linear system.

The paper [12] proposed the use of recurrent multi-layer N with external inputs NARX.

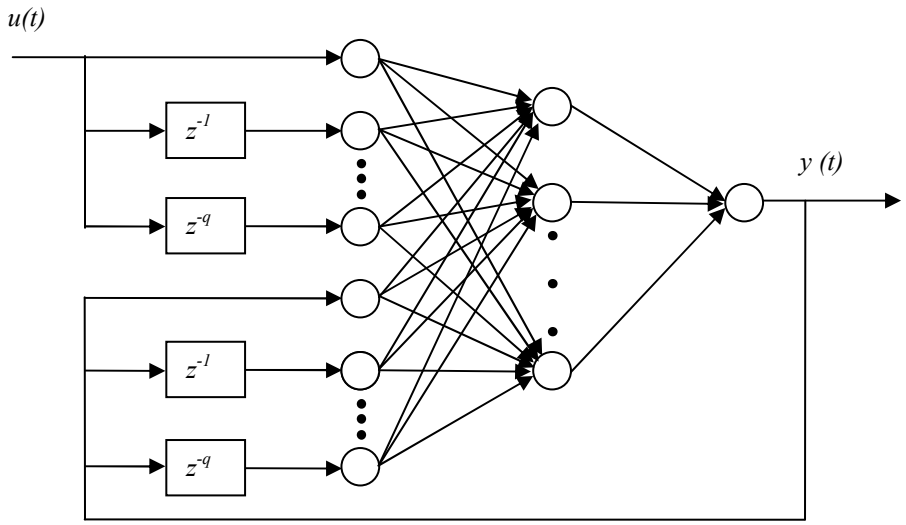


**Fig. 2.** The recurrent multilayer neural network NARX

The training model is given as

$$y(n+1) = f(y(n),...,y(n-q+1),u(n)),...,u(n-q+1)), \quad (2)$$

where *y(n)* is output vector, *u(n)* is input vector, *n* is the discrete time moment, *q* is power of the system.

Such a NN, which has feedback with single delays, allows constructing on its basis a model of a dynamic object of arbitrary complexity. Using this method requires veri-

fication of trained NF for adequacy with the use of new data not included in the training sample. Such NN is associated with the possibility of re-training NN [12].

The Matlab [13] Neural Network Toolbox application suite contains the most popular neurocontrollers (NPCs) with

- Neural Predictive Control (NPC),
- Nonlinear Auto Regressive Moving Average (NARMA-L2) model,
- Model Reference Controller (MRC).

In [14], a mathematical description of predictive neurorization using MATLAB system tools is presented. In [15], the NARMA-L2 controller is used for automatic control of the vessel on a variable course. When solving the problem of guidance and stabilization of the armament of a light armored machine, the NARMA-L2 neuro regulator is used in the contour of speed. As the authors note, NARMA-L2 acts as a relay regulator, whose output is switched to opposite limits, resulting in significant fluctuations in speed (up to 40% of the maximum). However, these neuro-regulators are not connected with physical model of object.

The purpose of this work is to build model and neuro-controller to control small satellite with default amount of reaction wheels.

## 3 Materials and methods

The main tasks of the paper are to create:

1. Simple simulator of satellite rotations, controlled by 3 or 4 reaction wheels, placed in different configurations. The simulation model will be configurable and easy to read.
2. An Artificial Intelligence (AI) learning module which will trigger the simulator and learn autonomously from the behavior of the simulated satellite, how to control its rotations.
3. The AI module, after trained for different configurations of wheels, will get commands with desired 3D rotation speeds and control the wheels to achieve the desired rotation.

### 3.1 Satellite simulator design

Simulator is developed using C++ programming language.

The satellite simulator is created to solve the next tasks:

- To provide physical model of physical object;
- To provide physical model of satellite with reaction wheels for rotation control;
- To provide possibility to control satellite using reaction wheels during simulation.

Simulator is divided to the such layers of logical implementations:

- Core of simulation,

- Satellite simulation.

Core is a general simulation that grants us encapsulated logic for creating and moving of material object. It also allows us to configure simulation and to log information about all objects in simulation. Satellite simulation extends material object logic with reaction wheels and physical facts (friction, gravity, gyroscope effect etc.).

The class diagram is given in Fig. 3.

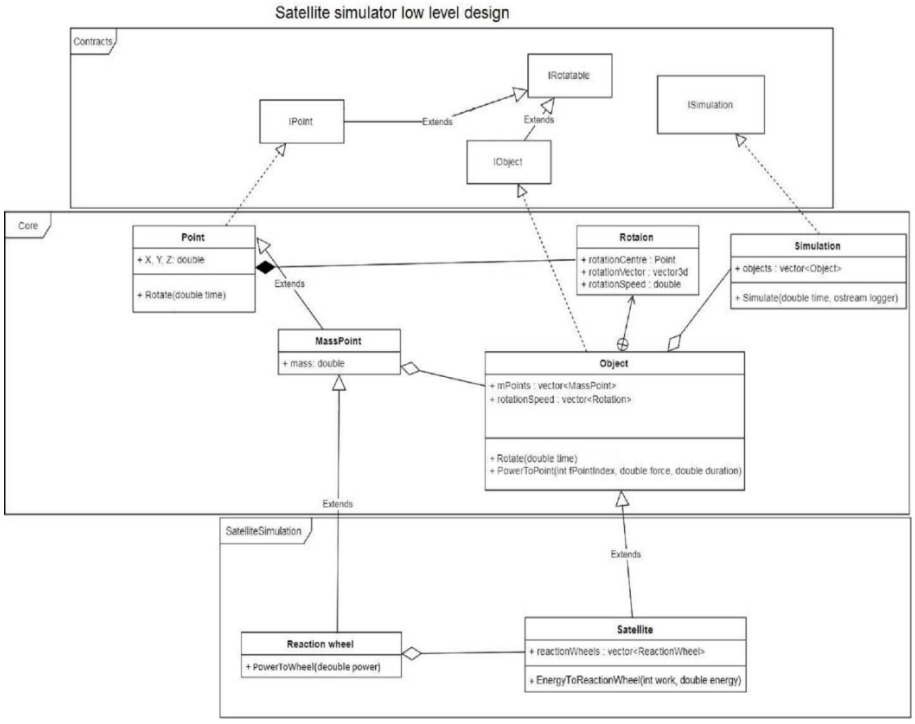


**Fig. 3.** Satellite simulator class diagram

Design layers:

- Contracts shows main entities of simulator and grants low coupling between their implementations. Contracts consists of abstractions;
- Core implements contracts. It contains primary physical model and Simulation entity.
- Satellite simulation extends Core with a dynamic of reaction wheel and satellite.

Entities:

- Point - provides an abstract point for further implementations;
- MassPoint - point which has mass and movement vector;
- Object - provides enumeration of points which interact with each other;

- ReactionWheel - inherited from MassPoint instances, is used for changing rotation speed of satellite by changing its angular momentum;
- Satellite - inherited from Object, provides simulated Satellite of arbitrary form, which moves and rotates using thrusters(ForcePoint) and reaction wheels;
- Simulator - provides enumeration of Object instances and configuration of scenario of their behavior.

The sphere in the Fig. 4 is a space, which limits the set of material points of the object. The center of mass is note center of the sphere, because its coordinates depends of coordinates and masses off other points.

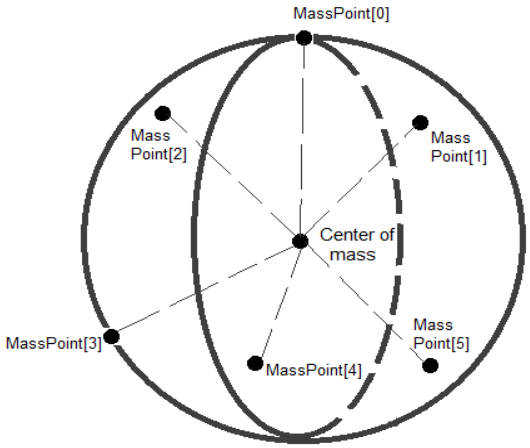


**Fig. 4.** The center of mass explanation

### 3.2 Neuro-controller design

The controller is created to solve the next tasks:

- Generate samples of satellite rotations.
- Train the Neural Network model, which can predict expected Energy on reaction wheel.
- Set the energy on reaction wheel based on training result.

During of training, the neural network must monitor and remember the dependence of the control signal $u(k\text{-}1)$ on the next value of the reaction of the control object that was before in the state $X(k\text{-}1)$. The values of the control signals and responses of the object are recorded and, on this basis, a training sample is formed.

$$U = \{P_i, T_i\}_{i=1}^{M} : P_i = \left[y(i)X(i-1)\right]^T, T_i = u(i) \tag{3}$$

We used and desired reaction.

In the training mode neural network must find and remember the dependence of control signal $u(k-1)$, in state before $S(k-1)$. When the object is controlled, the inverse neuro-emulator is connected as a controller and it is receiving the $rr(k)$ value from input $r(k+1)$:

$$rr(k) = \left[r(k+1)X(k)\right]^T. \tag{4}$$

The class diagram is given in Fig. 5.

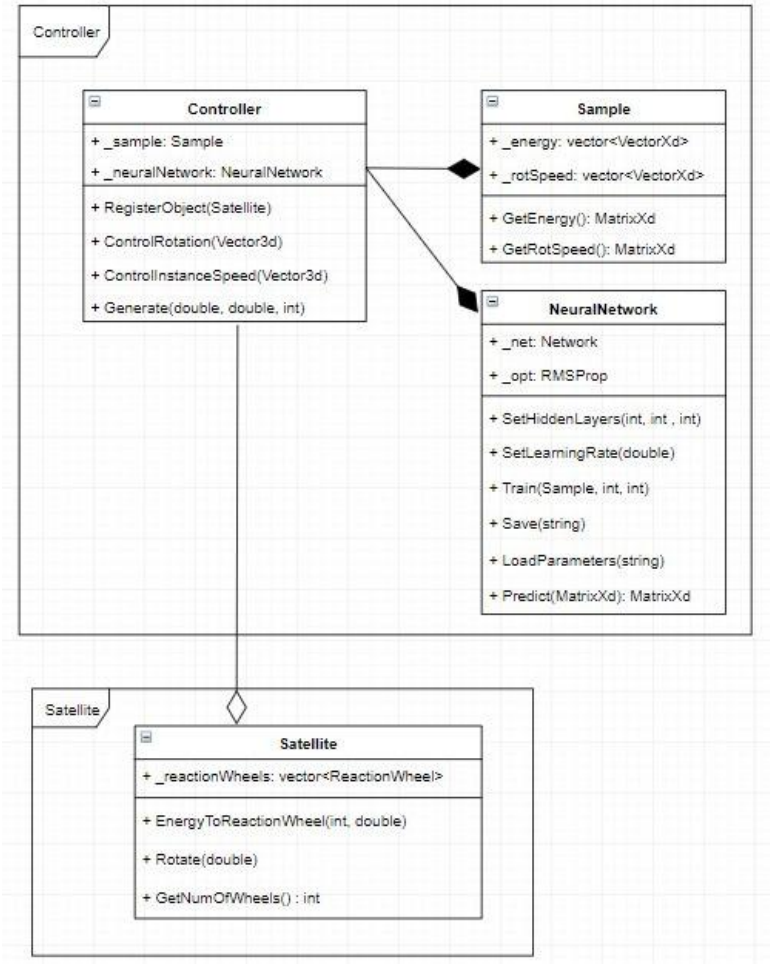


**Fig. 5.** Class controller diagram

Inputs in the control network is the satellite state (speed for each axes). The output is the control signal (torque) u(t). This is energy level for each rotation wheel.

We used mini-batch gradient descent algorithm for neural network training.

**The structure of neural network**

The neural network structure for this task looks like this:

- Input layer – 3 neurons (for speed by x,y,z),
- Hidden layer – 15 full-connected neurons with sigmoid activation function,
- Output layer – n neuron with predicted energy level, where n is equal amount of rotation wheels,
- The bias is used too.

The architecture of neuro-controller is chosen experimentally and given in Fig. 6.

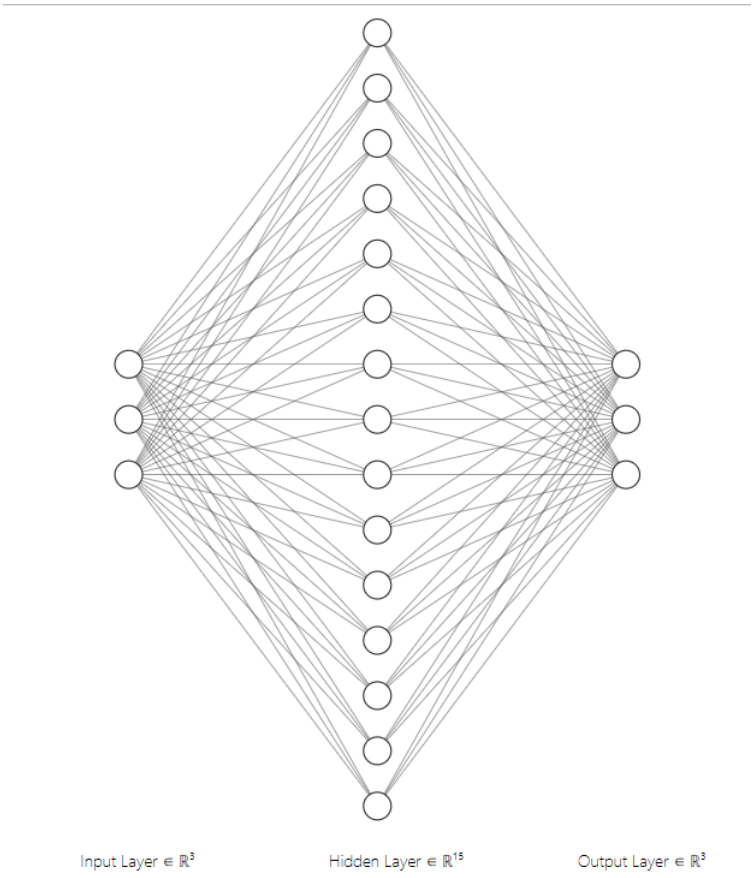


**Fig. 6.** Neural network architecture

We used mini-batch gradient descent in NN.

The goal of the algorithm is to find model parameters (e.g. coefficients or weights) that minimize the error of the model on the training dataset. It does this by making changes to the model that move it along a gradient or slope of errors down toward a minimum error value. This gives the algorithm its name of “gradient descent.”

Mini-batch gradient descent is a trade-off between stochastic gradient descent and batch gradient descent. In mini-batch gradient descent, the cost function (and therefore gradient) is averaged over a small number of samples, from around 10-500. This is opposed to the SGD batch size of 1 sample, and the BGD size of all the training samples.

Mini-batch gradient descent finally takes the best of both worlds and performs an update for every mini-batch of n training examples:

$$\theta=\theta-\eta\cdot\nabla\theta J(\theta;x(i:i+n);y(i:i+n)). \quad (5)$$

This allows us

- to reduces the variance of the parameter updates, which can lead to more stable convergence;
- can make use of highly optimized matrix optimizations common to state-of-the-art deep learning libraries that make computing the gradient w.r.t. a mini-batch very efficient. Common mini-batch sizes range between 50 and 256, but can vary for different applications.

# 4 Results

## 4.1 Stack of technologies

For neuro-controller realization we used

1. **Eigen** to provide vectors, matrixes, quaternions of different dimensions and working with them (it was mostly used in simulator) [16].
2. **MiniDnn** to provide neural network for creating controller of a satellite.

Parameters of NN is saved in NeuralConfig.h. These neural network parameters were chosen experimentally. We provided more than 500 training experiments with different neural network configuration. In the best attempts mean loss was be equal 0.013 and parameters there was:

```
NUMBEROFSAMPLES       1000
NUMBEROFHIDDENLAYERS     1
HIDDENLAYERSLENGTH     15
LEARNINGRATE         0.0007
BATCHSIZE         200
EPOCH             40000
```

**Table 1.** Training results

| # | BatchSize | Epoch | LearningRate | Loss function |
|---|---|---|---|---|
| 1 | 20 | 40000 | 0.0002 | 0.11 |
| 2 | 200 | 4000 | 0.0007 | 0.013 |
| 3 | 200 | 40000 | 0.002 | 0.008 |
| 4 | 200 | 40000 | 0.005 | 0.05 |
| 4 | 200 | 40000 | 0.001 | 0.07 |

Training time is appr 7 hours and 20 min. The computer configuration is given below:

```
Intel Core i3 (3,4 Ghz), 2 cores, NVidia GeForce, GT630, 2Gb
```

## 5 Conclusions

To sum up this article described how we could use neural networks for controlling satellites. Neural controllers is a very powerful method that allows us automate different processes and improve accuracy of its results.

An experimental study of the proposed criterion of 500 neuro-controllers was conducted, which showed its effectiveness compared to the traditional method (Loss function value is less than 0.05) of selecting neurotransmitters based on the least square root error method on the test data voter.

In the framework of further research, it is planned to test this criterion, along with other methods of neuro-control, which include the stage of preliminary neuro-identification of the control object: predictive model neuro management and hybrid neuro-PID control as well as using the Kalman cube filter.

## 6 References


1. Narendra, K. S., & Parthasarathy, K.: Identification and control of dynamical systems using neural networks. *IEEE Transactions on neural networks*, *1*(1), 4-27 (1990)
2. Prokhorov, D. V., & Wunsch, D. C.: Adaptive critic designs. *IEEE transactions on Neural Networks*, *8*(5), 997-1007 (1997)

3. Feldkamp, L. A., & Puskorius, G. V..:Training controllers for robustness: multi-stream DEKF. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)* 4, 2377-2382 (1994)
4. Prokhorov, D. V.: Toyota Prius HEV neurocontrol and diagnostics. *Neural Networks*, *21*(2-3), 458-465 (2008)
5. Haykin, S. S., Haykin, S. S., Haykin, S. S., Elektroingenieur, K., & Haykin, S. S.: *Neural networks and learning machines* (Vol. 3). Upper Saddle River: Pearson. (2009)
6. Kawafuku, R., Sasaki, M., & Kato, S.: Self-tuning PID control of a flexible micro-actuator using neural networks. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)* Vol. 3, 3067-3072 (1998).
7. Burakov, M. V., & Kurbanov, V. G.: Neuro-PID control for nonlinear plants with variable parameters. *ARPN Journal of Engineering and Applied Sciences*, *12*(4), 1226-1229 (2017).
8. Anderson, D. F., Ermentrout, B., & Thomas, P. J.: Stochastic representations of ion channel kinetics and exact stochastic simulation of neuronal dynamics. *Journal of computational neuroscience*, *38*(1), 67-82 (2015)
9. Zhernova, P. Y., Deineko, A. O., Bodyanskiy, Y. V., & Riepin, V. O.: Adaptive Kernel Data Streams Clustering Based on Neural Networks Ensembles in Conditions of Uncertainty About Amount and Shapes of Clusters. In *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)* 7-12 (2018)
10. Bodyanskiy, Y., Boiko, O., Zaychenko, Y., & Hamidov, G.: Evolving GMDH-neuro-fuzzy system with small number of tuning parameters. In *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)* 1321-1326 (2017)
11. Ramachandran, R., Madasamy, B., Veerasamy, V., & Saravanan, L.: Load frequency control of a dynamic interconnected power system using generalised Hopfield neural network based self-adaptive PID controller. *IET Generation, Transmission & Distribution*, *12*(21), 5713-5722 (2018)
12. Lee, C. C., Sheridan, S. C., Barnes, B. B., Hu, C., Pirhalla, D. E., Ransibrahmanakul, V., & Shein, K.:. The development of a non-linear autoregressive model with exogenous input (NARX) to model climate-water clarity relationships: reconstructing a historical water clarity index for the coastal waters of the southeastern USA. *Theoretical and Applied Climatology*, *130*(1-2), 557-569 (2017)
13. Medvedev V. S., Potjomkin V. G.: Nejronnye seti. MATLAB 6. Moscow, Dialog-MIFI, 496 p (2002)
14. Hwang, C. L., & Jan, C.: Recurrent-neural-network-based multivariable adaptive control for a class of nonlinear dynamic systems with time-varying delay. *IEEE transactions on neural networks and learning systems*, *27*(2), 388-401 (2016)
15. Yang, Y., Xiang, C., Gao, S., & Lee, T. H.: Data-driven identification and control of nonlinear systems using multiple NARMA-L2 models. *International Journal of Robust and Nonlinear Control*, *28*(12), 3806-3833 (2018)
16. Pukach, P., Il'kiv, V., Nytrebych, Z., Vovk, M., Shakhovska, N., & Pukach, P.: Galerkin Method and Qualitative Approach for the Investigation and Numerical Analysis of Some Dissipative Nonlinear Physical Systems. In *2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)* Vol. 1,. 143-146 (2018)