sciendo

VOLUME 18, N° 3 2024
Journal of Automation, Mobile Robotics and Intelligent Systems

Identification and Modeling of the Dynamical Object with the Use of HIL Technique

Submitted: 26th May 2023; accepted: 2nd February 2024

Łukasz Sajewski, Przemysław Karwowski

DOI: 10.14313/JAMRIS/3-2024/21

Abstract:
This article presents a comparison of a classical approach to identification of unstable object and an approach based on artificial neural networks. Model verification is carried out based on the Quanser Qube-Servo object with the use of myRIO real-time controller as the target. It is shown that model identification using neural networks gives a more accurate representation of the object. In addition, the hardware-in-the-loop (HIL) technique is discussed and used, for implementation of the control algorithm.

Keywords: *HIL, neural networks, inverted pendulum*

1. Introduction

Identification involves determining the temporal behavior of a system or process using measured signals, and the temporal behavior is determined within the classes of mathematical models. The main goal is to obtain the smallest possible error between the actual process or system and its mathematical model [1]. Modeling by using neural networks, although more complex, is often more accurate and allows us to better map the dynamics of the tested object. One of the basic issues in modeling of a real object is its validation. This is where the HIL technique comes to the rescue. Hardware-in-the-loop (HIL) simulation is a technique for testing embedded systems at the system level in a comprehensive and cost-effective manner. HIL is most often used for development and testing of embedded systems. A prerequisite for the use of this technique is that the testing can be accurately reproducible in the operating environments. HIL simulation requires a real-time simulation that models the individual components of the embedded system under test (SUT) and all relevant interactions within a given operating environment. The simulation monitors the SUT's output signals and forces synthetically generated input signals into the SUT at the appropriate time. The SUT's output signals are typically parameters set on the actuator and information displayed by the operator. Input signals to the SUT can include data read from sensors and parameters set by the operator. Outputs from the embedded system serve as inputs to the simulation, and the simulation generates outputs hat become inputs to the embedded system [2].

2. Inverted Pendulum

The rotating pendulum system is a classic system. It is most commonly used for teaching modeling and control. The designations used to model the QUBE-Servo rotary pendulum are shown in Figure 1 [3].

The rotary arm attached to the motor axis is denoted by the variable θ, while the pendulum attached to the end of the pivot arm is denoted by the angle α.

Note the following relationship:

- angle α is the angle with respect to the vertical position. Mathematically, this is determined by the formula:

$$\alpha = \alpha_{full} mod\ 2\pi - \pi, \tag{1}$$

where α_{full} is the angle of the pendulum as measured by the encoder;

- the movement of both angles is positive if the movement is counterclockwise (CCW) and applying a positive voltage to the motor causes counterclockwise rotation.

The rotating axis of the arm is connected to the QUBE-Servo system. The arm has length L_r, moment of inertia J_r. The servo arm should rotate counterclockwise when the control voltage is positive.

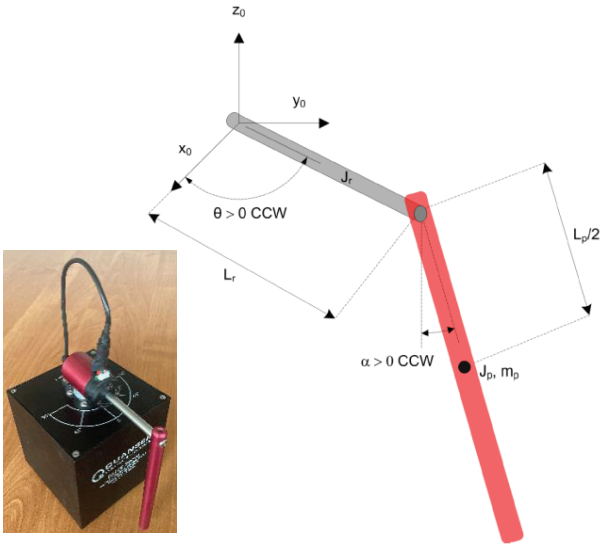

Figure 1. Model conventions of rotary pendulum and the real plant [3]

The link of the pendulum is connected to the end of the rotating arm. Its total length is L_p, and its center of mass is at the point $l = \frac{L_p}{2}$. The moment of inertia with respect to the center of mass is J_p.

The angle α of the rotary pendulum takes the value of zero when it is pointed vertically downward. Counterclockwise motion results in increasing values of the rotation angle.

The equations of motion (EOM) for the pendulum system were developed using the Euler-Lagrange method. This method is most often used when modeling complex systems-for example, robot manipulators with multiple joints. This gives the total kinetic and potential energy of the system under study. Then the derivatives are calculated to find the equations of motion. The resulting nonlinear EOMs are [3]:

$$(J_r + J_p \sin\alpha^2)\ddot{\theta} + m_p l L_r \cos\alpha\ddot{\alpha} + 2J_p \sin\alpha\cos\alpha\dot{\theta}\dot{\alpha} - m_p l L_r \sin\alpha\dot{\alpha}^2 = \tau - b_r\dot{\theta} \quad (2)$$

and

$$J_p\ddot{\alpha} + m_p l L_r \cos\alpha\ddot{\theta} - J_p \sin\alpha\cos\alpha\dot{\theta}^2 + m_p g l \sin\alpha = -b_p\dot{\alpha} \quad (3)$$

where $J_r = \frac{m_r r^2}{3}$ is the moment of inertia of the rotary arm with respect to the pivot (i.e. rotary arm axis of rotation) and $J_p = \frac{m_p L_p^2}{3}$ is the moment of inertia of the pendulum link with respect to the pendulum's axis of rotation (i.e., the pendulum's axis of rotation). The viscous damping acting on the pivot arm and the pendulum link is b_r and b_p, respectively. The torque generated by the servomotor at the base of the pivot arm is

$$\tau = \frac{k_m}{R_m}(v_m - k_m\dot{\theta}). \quad (4)$$

When the nonlinear EOM are linearized about the operating point, the resultant linear EOM for the rotary pendulum are defined as [3]:

$$J_r\ddot{\theta} + m_p l L_r\ddot{\alpha} = \tau - b_r\dot{\theta} \quad (5)$$

and

$$J_p\ddot{\alpha} + m_p l L_r\ddot{\theta} + m_p g l\alpha = -b_p\dot{\alpha} \quad (6)$$

Solving for the acceleration terms yields:

$$\ddot{\theta} = \frac{1}{J_t}(m_p^2 l^2 L_r g\alpha - J_p b_r\dot{\theta} + m_p l L_r b_p\dot{\alpha} + J_p\tau) \quad (7)$$

and

$$\ddot{\alpha} = \frac{1}{J_t}(-m_p g l J_r\alpha + m_p l L_r b_r\dot{\theta} - J_p b_p\dot{\alpha} - m_p l L_r\tau), \quad (8)$$

where

$$J_t = J_p J_r - m_p^2 l^2 {L_r}^2. \quad (9)$$

Since we are dealing with nonlinear unstable dynamical system, the practical identification process is much more complicated.

Symbol	Description	Value
DC Motor		
V_{nom}	Nominal input voltage	18.0 V
τ_{nom}	Nominal torque	22.0 mN-m
ω_{nom}	Nominal speed	3050 RPM
I_{nom}	Nominal current	0.540 A
R_m	Terminal resistance	8.4 Ω
k_t	Torque constant	0.042 N-m/A
k_m	Motor back-emf constant	0.042 V/(rad/s)
J_m	Rotor inertia	4.0×10^{-6} kg-m^2
L_m	Rotor inductance	1.16 mH
m_h	Module attachment hub mass	0.016 kg
r_h	Module attachment hub radius	0.0111 m
J_h	Module attachment moment of inertia	0.6×10^{-6} kg-m^2
Inertia Disc Module		
m_d	Disc mass	0.053 kg
r_d	Disc radius	0.0248 m
Rotary Pendulum Module		
m_r	Rotary arm mass	0.095 kg
L_r	Rotary arm length (pivot to end of metal rod)	0.085 m
m_p	Pendulum link mass	0.024 kg
L_p	Pendulum link length	0.129 m
Motor and Pendulum Encoders		
	Encoder line count	512 lines/rev
	Encoder line count in quadrature	2048 lines/rev
	Encoder resolution (in quadrature)	0.176 deg/count
Amplifier		
	Amplifier type	PWM
	Peak Current	2 A
	Continuous Current	0.5 A
	Output voltage range	±10 V

Figure 2. Quanser Qube-Servo parameters [3]

In Figure 2, we have Quanser Qube-Servo parameters given by the manufacturer.

Using these parameters and EOMs we can write a linearized model of an object by the following equations [6]:

$$\begin{cases}\dot{x}(t) = Ax(t) + Bu(t)\\ y(t) = Cx(t) + Du(t)\end{cases} \quad (10)$$

where

$$x(t) = [\theta(t)\alpha(t)\dot{\theta}(t)\dot{\alpha}(t)]^T \quad (11)$$

$$y(t) = [\theta(t)\alpha(t)]^T \quad (12)$$

and

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{m_p^2 l^2 L_r g}{J_t} & -\frac{b_r J_p}{J_t} & -\frac{m_p b_p l L_r}{J_t} \\ 0 & \frac{m_p l J_r g}{J_t} & -\frac{m_p b_r l L_r}{J_t} & -\frac{b_p J_r}{J_t} \end{bmatrix} \quad (13)$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{J_p}{J_t} \\ \frac{m_p l L_r}{J_t} \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad (14)$$

where b_r is equivalent viscous damping coefficient rotary ((N*m*s)/rad), b_p is equivalent viscous damping coefficient pendulum ((N*m*s)/rad).

By add actuator dynamics

$$A(3,3) = A(3,3) - \frac{k_m^2}{R_m} * B(3,1) \quad (15)$$

$$A(4,3) = A(4,3) - \frac{k_m^2}{R_m} * B(4,1) \quad (16)$$

$$B = \frac{k_m}{R_m} * B \quad (17)$$

Figure 3. Connection diagram [3]

ultimately we get

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1.520e+02 & -12.254 & -0.500 \\ 0 & 2.643e+02 & -12.112 & -0.870 \end{bmatrix} \quad (18)$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 50.637 \\ 50.048 \end{bmatrix} C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \quad (19)$$

The pools are the following

$$\begin{bmatrix} 0,000 \\ 15,672 \\ -16,943 \\ -0,957 \end{bmatrix}, \quad (20)$$

which confirms that the considered system is unstable.

3. Real Time Controller

NI myRIO real-time controller has been used to manage the Quanser Qube-Servo object. This controller is a portable reconfigurable I/O (RIO) device that can be used to design control, robotics, and mechatronics systems [4]. Encoder signals and a motor control signal were connected to the myRIO. MATLAB/SIMULINK software has been used as the development environment along with the Quanser QUARC add-on, giving the possibility to control a plant with the use of real-time target (QUARC Linux RT ARMv7 Target). QUARC^{TM} is the most efficient way to create real-time applications on hardware. QUARC generates real-time code directly from drivers programmed with Simulink. It runs the program on the target device in real time [3]. This approach allows us to compile the code using a PC and then run it on a real-time controller (myRIO), which is connected directly to the plant. When RT Target is started, the PC is used only for the presentation of process variables. The connection diagram is shown in Figure 3.

In the discussed task, the following hardware and software have been used:

- Komputer PC
  - Processor: Intel(R) Core i5-12600 3,3GHz,
  - RAM: 32GB,
  - System: Microsoft Windows 11 Pro
- MATLAB Version: 9.11.0.2022996 (R2021b),
  - SIMULINK version 10.4,
  - SIMULINK Coder version 9.6,

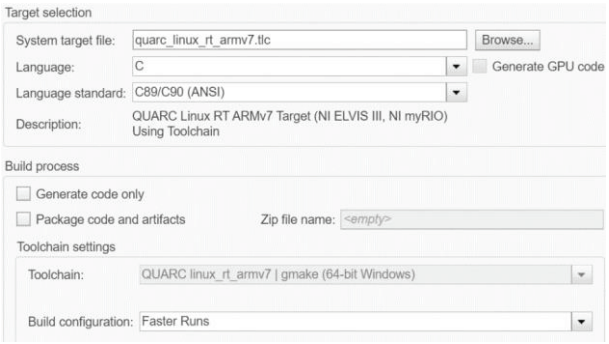

Figure 4. MATLAB/SIMULINK real time target settings

  - MATLAB Coder version 5.3.
- Quanser QUARC 2021 SP1
- NI MyRio-1900:
  - Xilinx Z-7010 processor with 2 cores,
  - Processor speed is 667MHz.

Full state feedback method has been used to stabilize the system, and the LQR technique has been used to determine the matrix K stabilizing the system.

New poles location is the following

$$\begin{bmatrix} -70 \\ -10 \\ -5 \\ -5 \end{bmatrix}. \quad (21)$$

New poles location has been taken based on tests on real plant. Figure 5 presents the MATLAB/SIMULINK control schema, where QUARC HIL blocks have been used. Having the stabilized system at hand, we can compare step responses of the real plant and the model created by the use of EOMs and factory data. The step response of the systems can be seen in Figures 6 and 7.

As can be seen, the response of the simulated system (black) and the real object (blue) are not equal, this is due to a small deviation to one side of pendulum and rotor (among other factors).

Since the EOMs model is not precise, we will proceed with the construction of the neural model. To do this, it is necessary to record the response of the real system to a specially prepared input signal. The next step is to train the network. A nonlinear autoregressive neural network with external input (NARX) was selected to train the network. This type of network is useful to predict time series data. The network had 2 hidden layers, each with 10 neurons. The network had 10 backward samples of the forcing signal and 10 backward samples of the feedback signal as input arguments. The signal, which is used to train the network, is shown in Figure 8.

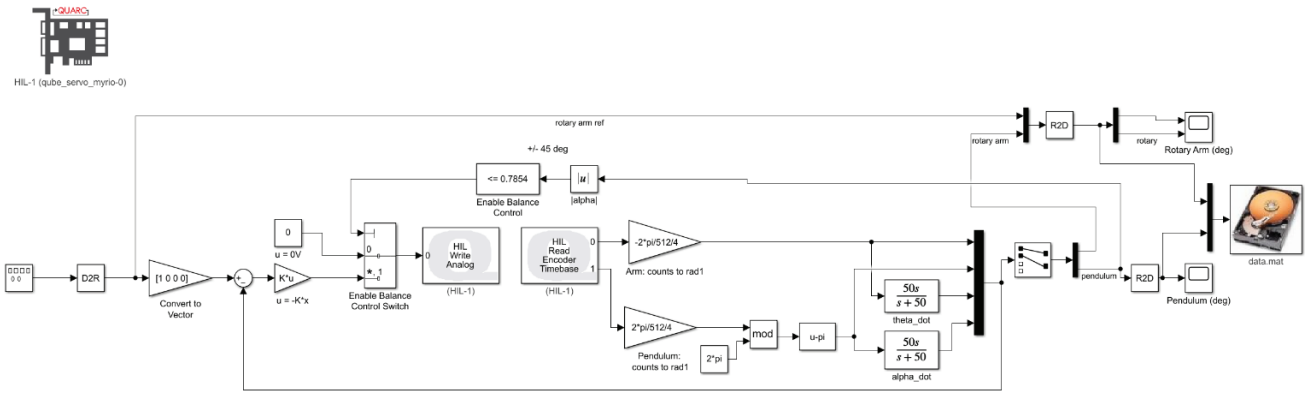

Figure 5. Schematic modal control (loop time set for 0.01s)

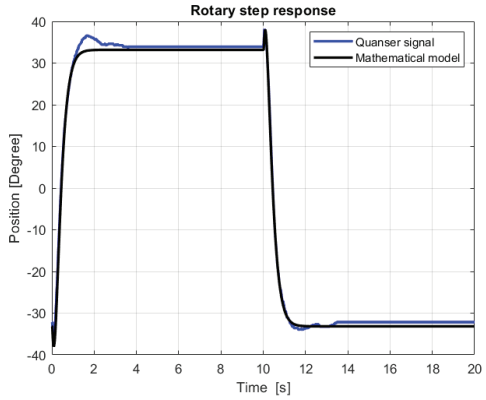

Figure 6. Rotary Step response

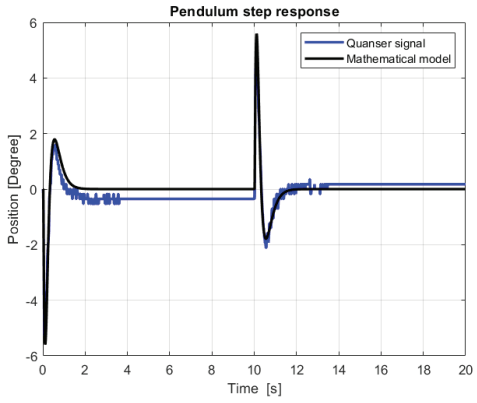

Figure 7. Pendulum step response

4. Modeling Results

Figure 9 shows the schematic program that has been implemented into the myRIO controller. The forcing signal from the signal generator is sent to both the Quanse Qube-Servo object and the mathematical and neural model. The responses are then shown in the Scope window.

During the measurements, it was necessary to verify the loop time of the myRIO controller. Initially this parameter has been set to 0.01[s]. Figure 10 shows the actual loop time during the test.

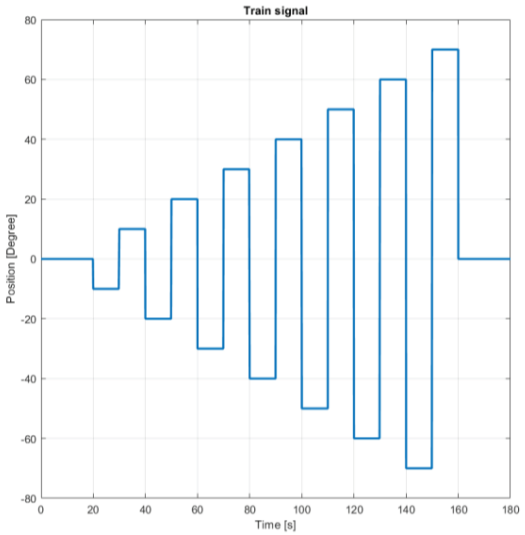

Figure 8. Training signal

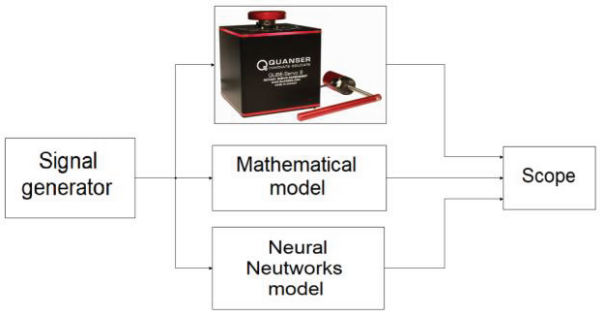

Figure 9. Wiring Diagram [3]

Figures 11 and 12 compare rotary and pendulum responses between measurement, mathematical model, and Neural-Networks model. In Figures 13 and 14 we can see the error between the real object signals and those generated from models.

The performance assessment by the use of integral absolute error (IAE) criteria have been given in Table 1.

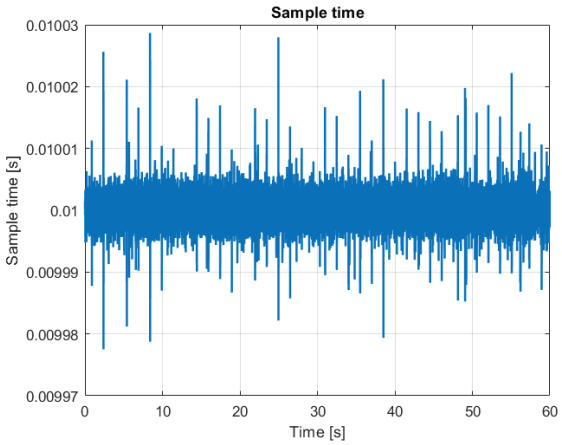

Figure 10. Loop time during the run

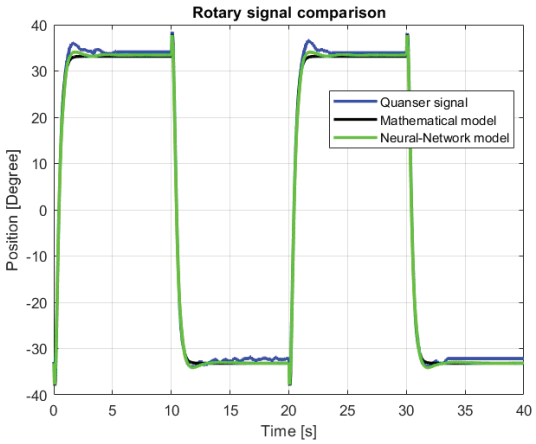

Figure 11. Rotary signal comparison

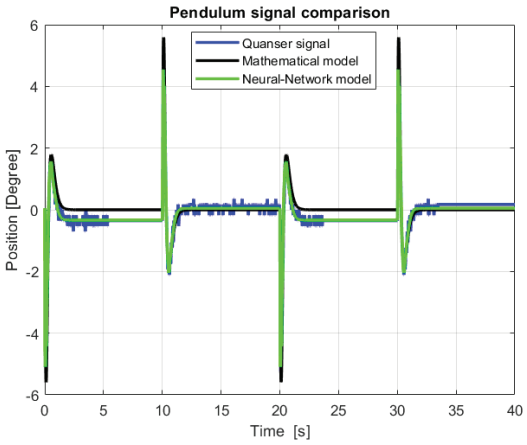

Figure 12. Pendulum signal comparison

Table 1. Integral absolute error criterion

Object \ Response	Factory given model	Tuned Neural Networks model
Pendulum	4.46	1.704
Rotary	14.73	11.73

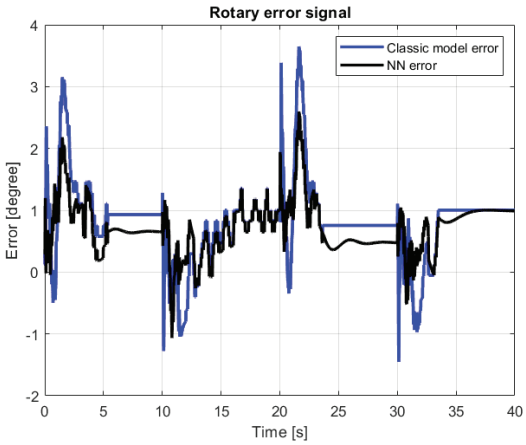

Figure 13. Rotary error

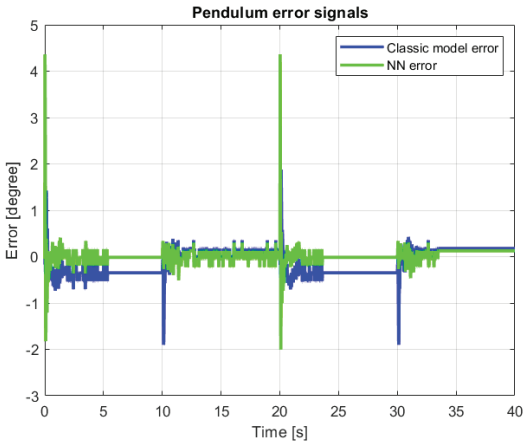

Figure 14. Pendulum error

5. Concluding Remarks

In the case of the Quanser Qube Servo device, the pendulum encoder cable acted like a spring pushing the arm away, thus interfering with the movements. In the step response (Figs. 6 and 7), it can be seen that the tilt in each direction is not perfectly symmetrical: this was related to the mentioned cable. From the error signals plot, it can be seen that a properly trained network gives much more accurate results than a mathematical model. Since the plant is nonlinear, it is crucial to choose the appropriate testing signal for Neural Network training.

Using the HIL technique, it is important to remember the limitations of the target hardware (in this case, myRIO). When implementing control algorithms to target hardware, it is necessary to check real HIL loop time since it can influence the quality of the control.

Having a good quality mathematical model opens up a lot of possibilities when designing algorithms to control an object. For the mathematically determined model, it is necessary to select the appropriate hardware gain, which results from the use of electronic components in the object—for example, an encoder or a motor.

AUTHORS
Łukasz Sajewski* – Faculty of Electrical Engineering, Bialystok University of Technology, Bialystok, 15-351, Poland, e-mail: l.sajewski@pb.edu.pl.
Przemysław Karwowski – Faculty of Electrical Engineering, Bialystok University of Technology, Bialystok, 15-351, Poland, e-mail: pkarw1@wp.pl.
*Corresponding author

ACKNOWLEDGEMENTS

The studies have been carried out in the framework of work No. WZ/WE-IA/5/2023 and financed from the funds for science by the Polish Ministry of Science and Higher Education.

References

[1] R. Isermann, M. Münchhof, *Identification of Dynamic Systems: An Introduction with Applications*, Springer-Verlag Berlin Heidelberg 2011. 10.1007/978-3-540-78879-9.

[2] J. Ledin, "Simulation Engineering: Build Better Embedded Systems Faster", *CRC Press* 2001. doi: 10.1201/9781482280722.

[3] J. Apkarian, M. Lévis, Quanser Student Workbook QUBE-Servo Experiment for MATLAB/Simulink Users, Markham, Ontario, 2014.

[4] NI myRIO-1900 User Guide and Specifications, National Instruments, 2018.

[5] L. Zadeh, "Probability measures of fuzzy events", *Journal Math. Analysis and Appl.*, vol. 23, no. 2, 1968, 421–427. doi: 10.1016/0022-247X(68)90078-4.

[6] W. Rudin, "Principles of mathematical analysis", McGraw-Hill: New York, 1967, 10–54.

[7] T. Gabor, S. Illium, M. Zorn, C. Lenta, A. Mattausch, L. Belzner, C. Linnhoff-Popien, "Self-Replication in Neural Networks", *Artif Life*, vol. 28, no. 2, 2022, 205–223. doi: 10.1162/artl_a_00359.