MATLAB TOOLBOX FOR ADAPTIVE IDENTIFICATION WITH GRADIENT DESCENT ALGORITHM

Murat ÜNVER[1], Peter M. BENEŠ[2]

[1, 2] Department of Instrumentation and Control Engineering, Czech Technical University in Prague

{murat.unver, petermark.benes}@fs.cvut.cz

Abstract:

This paper presents a new MATLAB Simulink Toolbox for adaptive identification with use of high order neural units (HONUs) and gradient descent (GD) based learning. The toolbox allows users to investigate the potentials for adaptive identification via HONUs for dynamic modeling of linear and weakly non-linear industrial processes. The key contribution of this work is the development of a new toolbox for implementation of real-time identification and extension for adaptive control in the MATLAB Simulink framework.

Keywords:

adaptive identification, gradient descent (GD), higher order neural unit (HONU), linear neural unit (LNU), quadratic neural unit (QNU)

1 Introduction

This paper presents a new toolbox implemented in MATLAB Simulink for adaptive identification via dynamic linear neural units (LNU) and dynamic quadratic neural unit (QNU) with use of the famous gradient descent algorithm (GD). Till now, HONUs have presented numerous successful results in both theoretical and well and real-time engineering applications [1]–[4]. As an initial, this paper recalls the key structures of HONU architectures for adaptive identification. Following this, the gradient decent (GD) algorithm with application to dynamic HONUs, as most recently presented in [5] is recalled. Then, this paper presents implementation of the HONU toolbox for real-time adaptive identification with connotations to the CTU Roller Rig application as a continuation of the works [4], [6].

2 Structure of Neural Unit (LNU and QNU)

The toolbox is created for two different neural unit structures; LNU (HONU, r=1) and QNU (HONU, r=2). The general form of an LNU may be given as follows

$$\tilde{y} = \sum_{i=0}^{n} x_i w_i = w_0 . x_0 + w_1 . x_1 + \ldots + w_n . x_n = \mathbf{w} \cdot col^{r=1}(\mathbf{x}) \tag{1}$$

where r denotes the polynomial order, $\mathbf{x}$ is a vector of inputs and $\mathbf{w}$ is updatable vector of neural weights. Similarly a second order HONU architecture, more explicitly termed as a QNU (HONU, r=2) may be given as

$$\tilde{y} = \sum_{i=0}^{n} \sum_{j=0}^{n} x_i x_j w_{i,j} = w_{0,0} x_0 x_0 + w_{0,1} . x_0 x_1 + \ldots + w_{n,n} . x_n{}^2 = \mathbf{w} \cdot col^{r=2}(\mathbf{x}) \tag{2}$$

Where the long column vector form $col^{r=1}(\mathbf{x})$ comprises of previous step-delayed outputs $\tilde{y}(k-1)$ and further previous process inputs $u(k-1)$, more explicitly expressed as

$$col^{r=1}(\mathbf{x}) = \{x_i ; i = 0..n_x\}, \tag{3}$$

Further for a quadratic neural unit (QNU, i.e. HONU r=2), the long column vector maybe analogically expressed as

$$col^{r=2}(\mathbf{x}) = \{x_i x_j; i = 0..n_x, j = i..n_x\}. \tag{4}$$

In case of dynamic linear unit, $col^{r=i}(\mathbf{x})$ step-delayed HONU model outputs, are incorporated however in the sense of static HONUs, the real process outputs may be chosen instead. This further simplifies the recurrent derivatives which may be seen in the proceeding section regarding the derived gradient descent update rule.

3 Gradient Descent (GD) Learning Algorithm for HONU Adaptive Identification

The toolbox created as purpose of this project, gradient descent algorithm is used. It is one of the fundamental algorithm behind neural unit identifier and controller applications. General form of the algorithm is adopted to LNU and QNU. It is expressed for LNU as below;

$$\Delta\mathbf{w} = -\frac{\mu}{2}\cdot\frac{\partial e^2(k)}{\partial\mathbf{w}} = -\mu\cdot e(k)\cdot\frac{\partial(y_r(k)-\tilde{y}(k))}{\partial\mathbf{w}} = \mu\cdot e(k)\cdot\frac{\partial\tilde{y}(k)}{\partial\mathbf{w}} \tag{5}$$

where μ represents learning rate of the weight adaptation, $e(k)$ represents current error between real and neural unit output, and $\partial y(k)\partial wi$ corresponds to the computed partial derivative of y with respect to each neural weight. Often for practical engineering applications, a modification of the classical rule yields via normalization of the learning rate. This modification is termed more explicity as the normalized gradient decent rule (NGD) which may be expressed as

$$\Delta\mathbf{w} = \frac{\mu}{\left\|col^{r=1}(\mathbf{x})\right\|_2^2 + 1}\cdot e(k)\cdot\frac{\partial\tilde{y}(k)}{\partial\mathbf{w}}. \tag{6}$$

Once the weight update rule is established, each previous set of neural weights may be updated via the general update law. This form is also applicable in the sense of batch training algorithms.

$$\mathbf{w}(k)=\mathbf{w}(k-1)+\Delta\mathbf{w}. \tag{7}$$

4 Block Diagram of Adaptive Identification via HONUs

Above LNU, QNU, and GD concepts are explained in detail, here block diagram of adaptive identification is illustrated.

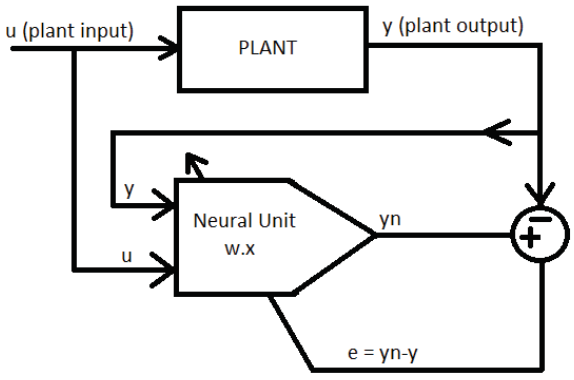

Fig. 1: Block Diagram of Adaptive Identification.

Process of identification can be described as, input signals inserted to system, as result output signal comes. These signals u and y are used to created vector of **x** as input vector and weights are updated by error calculation between system output and neural output. According to sampling rate these process repeat until neural unit identifies and learn the plant. QNU learns faster than LNU, therefore error rate of QNU is less than LNU. These comparison is shown in next section.

5 Matlab Simulink Toolbox

In this part, toolbox main structure, its function, and other details are presented with figures and explanations.Toolbox includes 5 main parts; system definition, **x** vector definition, LNU adaptive identifier, QNU adaptive identifier and visualization. In addition, there is also small script for parameters like time delay and initial conditions of neural weights. Parameters script given in appendix.

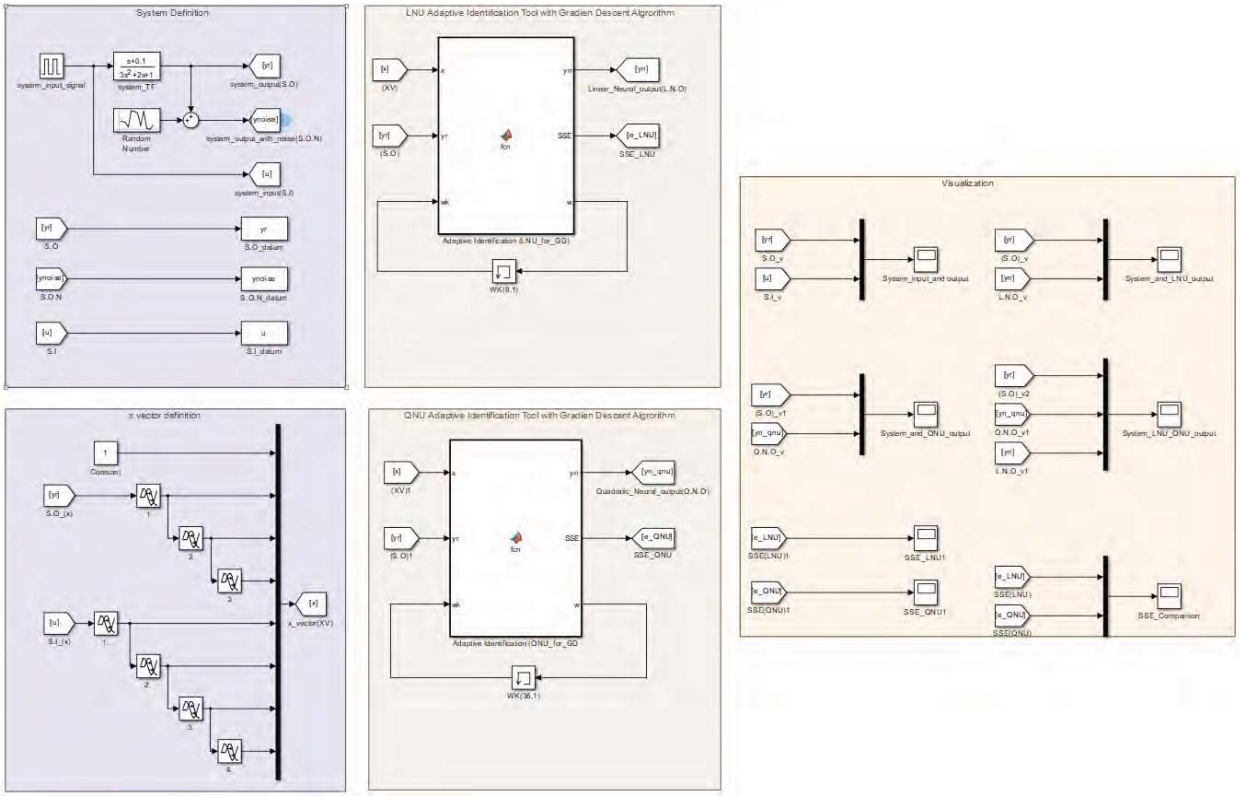

Fig. 2: MATLAB Simulink Toolbox.

In the following sections, toolbox sub tools are explained in detail.

5.1 System Definition

System definition part is created for users to insert their plant. For demonstration, the theorectical plant of second order is chosen as

$$\frac{s+0.1}{3s^2+2s+1} \tag{8}$$

The implementation is illustrated in Fig. 3;

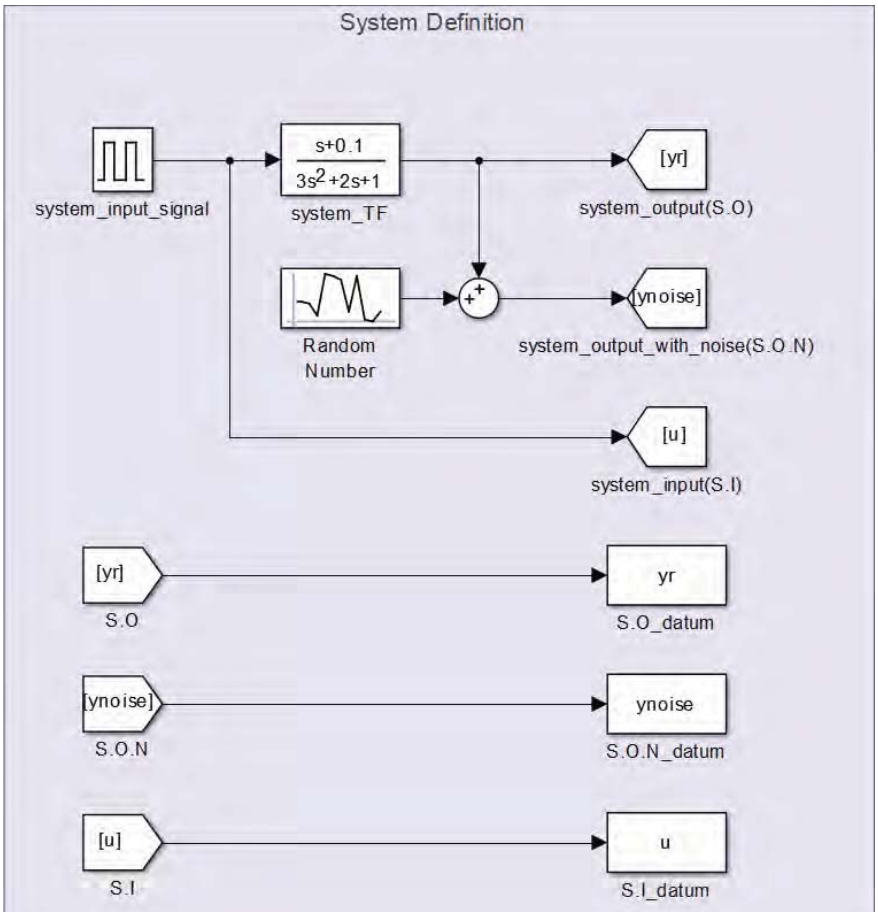

Fig. 3: System Definition.

Where u represents system input. Input signal is created by pulse generator for simulation. System output is assigned as "y_r" which means real system output. Also output signal with noise is add system for better understanding of adaptive identifier. All datum is connected to sink block to prevent chaos in the toolbox. And all datum are exported to workspace for observation in case of need.

5.2 Neural Input Vector Definition

As it is mentioned in previous sections, **x** is input vector for neural unit. It has components "*u*" and "*y*". After studying on sample and different systems, amount of "*u*" is chosen as 4, "*y*" is chosen as 3. Later these numbers are presented as $n_u = 4$, and $n_y = 3$.

Sampling of input datum is achieved by transport delay. Inside the configuration time delay is defined as dT to make users change this rate according to their applications.

Block of **x** vector definition is shown below.

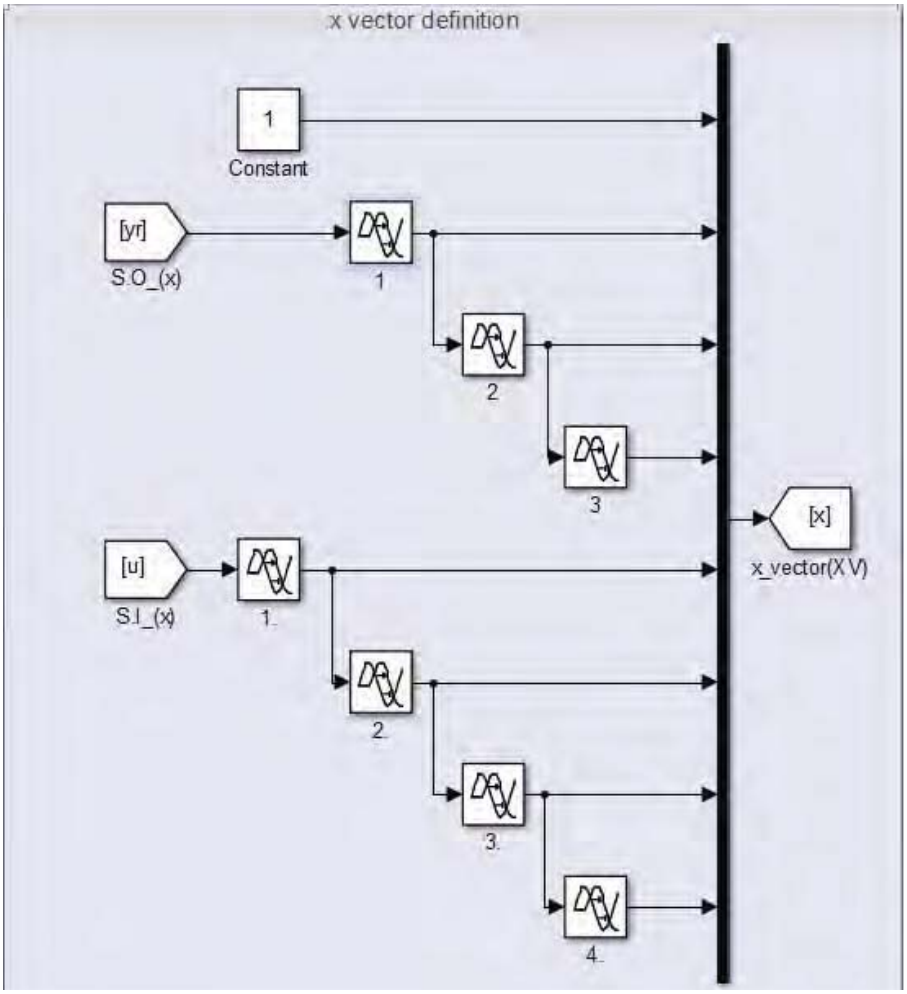

Fig. 4: Input Vector "**x**" Structure.

As it is visible in block, "**u**" and "**y**" signal come to the transport delay modules with sink blocks. **x** vector is inserted to another sink block to connect the MATLAB function module as input. Transport delay configuration creates vector as

$$\mathbf{x}(k) = [1 \quad \tilde{y}(k) \quad \tilde{y}(k-1) \quad \tilde{y}(k-2) \quad u(k) \quad u(k-1) \quad u(k-2) \quad u(k-3)] \tag{9}$$

Next parts mentions main tool for toolbox, LNU, and QNU adaptive identification tool.

5.3 LNU Adaptive Identification Tool

This part represents one of main tool for toolbox. Signals and input vector which are defined in previous tools are used as input for tool. These signals are processed inside the tool with embedded script. As result, tool gives 3 outputs there are; y_n (neural output), w (neural weights), and SSE (Sum of Square Error). The tool is illustrated as following;

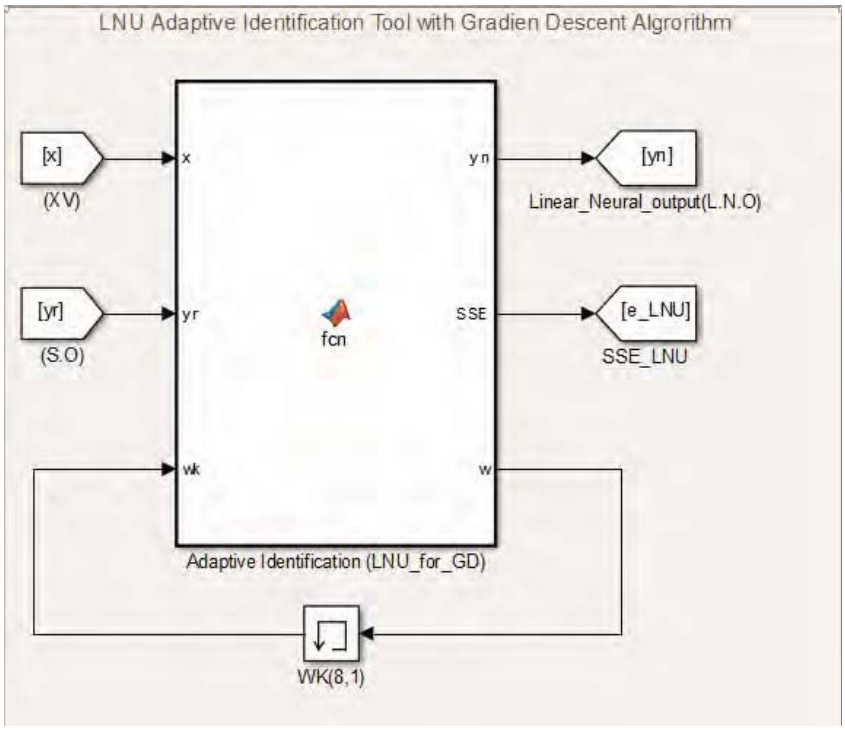

Fig. 5: LNU Adaptive Identification Tool.

Another variable in this tool "**w**(k)". This variable updates itself in every cycle. For first cycle its initial value is zero. However, these initial conditions can vary so **w**(k) is assigned as variable. As result, user can change this value in parameter section according to need.

5.4 QNU Adaptive Identification Tool

This section mentions another main tool for toolbox. Signals and input vector which are defined in previous tools are used as input for tool. These signals are processed inside the tool with embedded script. As result, tool gives 3 outputs there are; y_n (neural output), **w** (neural weights), and SSE (Sum of Square Error). The tool is illustrated in Fig. 6. Another variable in this tool "**w**(k)". This variable updates itself in every cycle. For first cycle its initial value is zero. However this initial conditions can vary so **w**(k) is assigned as variable. As result, user can change this value in parameter section according to need.

Remark: Dimension of weights vector is greater than LNU, due to **x** *vector. For better understanding, readers may refer to the neural unit structure section.*

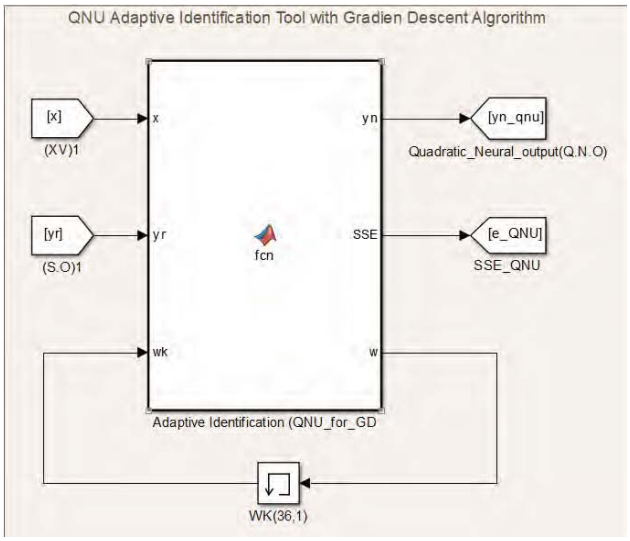

Fig. 6: QNU Adaptive Identification Tool.

5.5 Visualization

Visualization tool allows user to observer and compare results. This part includes; main system input and output signal graph, LNU output with system output graph, QNU output with system output graph, both LNU and QNU output with system output, SSE results, and SSE results for QNU and LNU for comparison. Tool is shown as below.

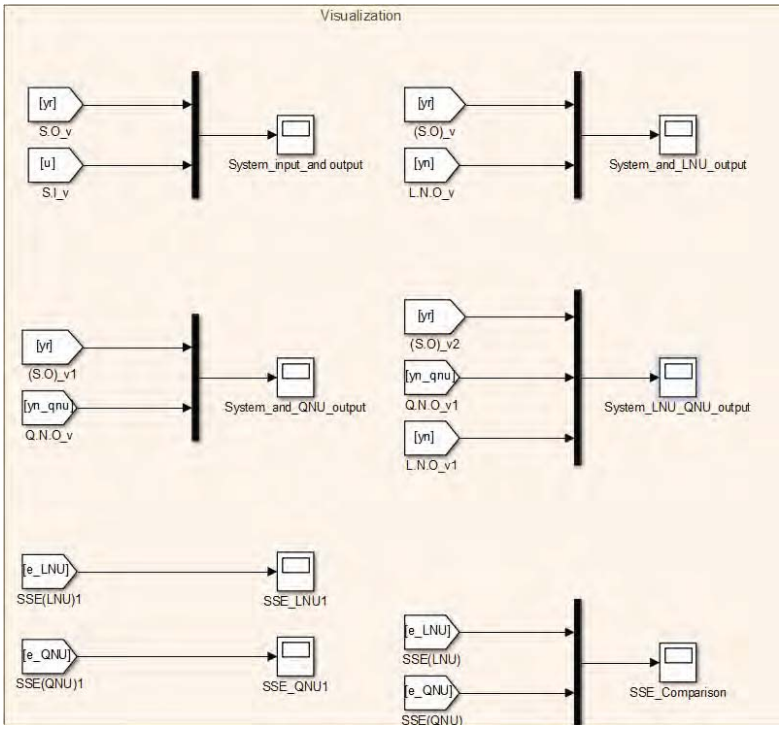

Fig. 7: Visualization.

For visualization scope tool of Simulink is used. In the following section results of example system are given.

6 Toolbox Results For Sample System

This section mainly illustrates result of the system chosen as example. Following figure shows system input (u) and output (y).

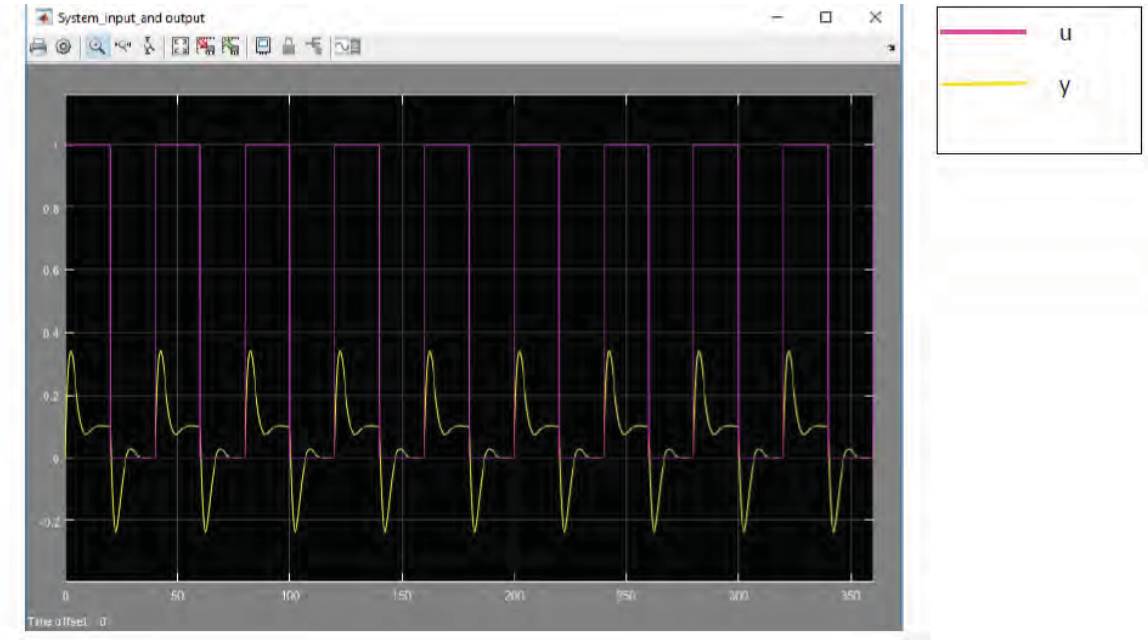

Fig. 8: System Input and Output.

Next figure illustrates system output with LNU output (y_n).

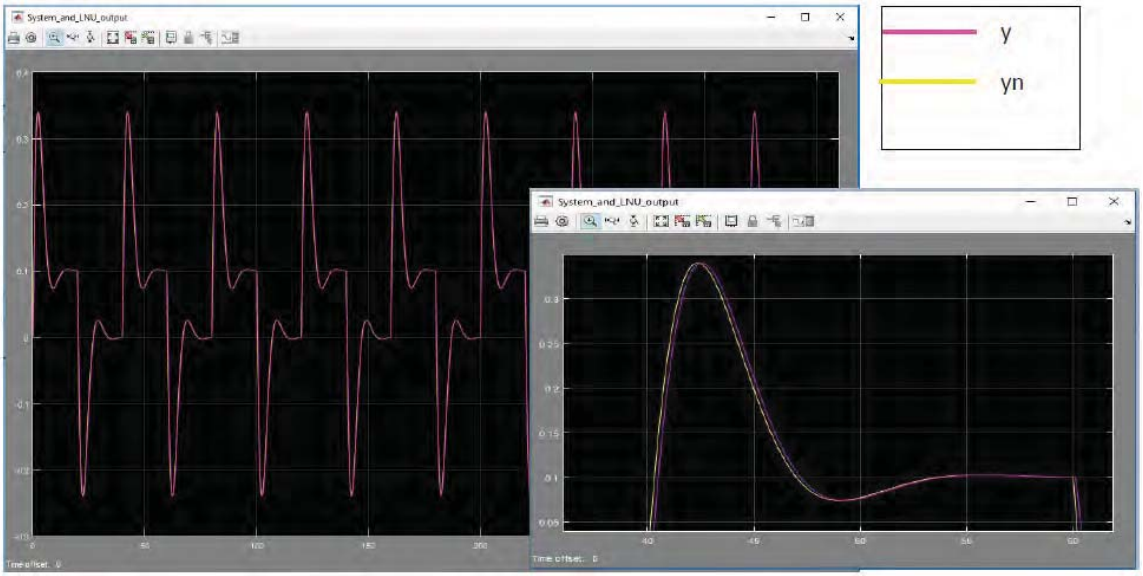

Fig. 9: System Output and LNU Output

Following figure shows system output with QNU output (y_n_QNU).

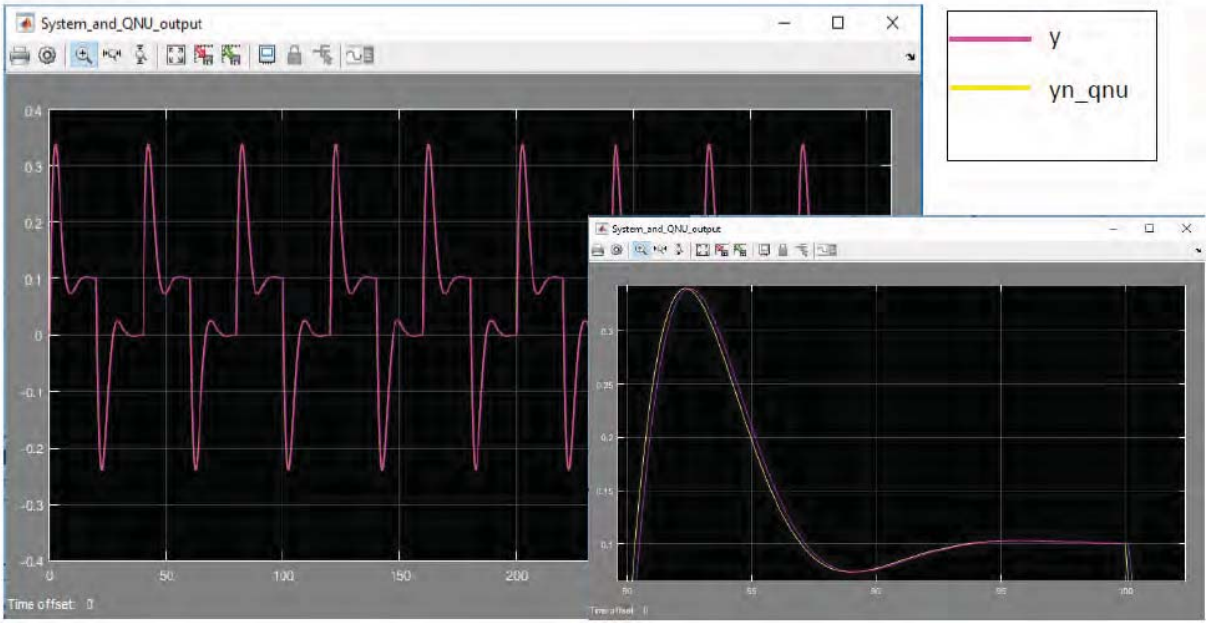

Fig. 10: System Output and QNU Output.

Next figure includes system output, LNU output, and QNU output together for easy comparison.

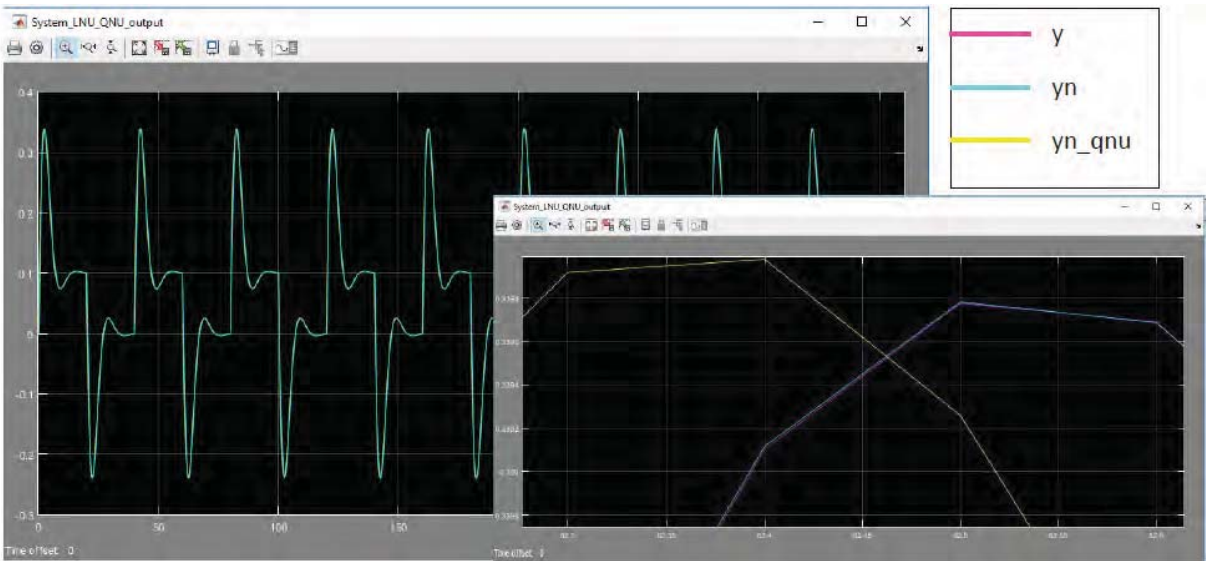

Fig. 11: System Output, LNU Output and QNU Output

Next graphs show SSE with respectively LNU, QNU, and LNU-QNU together.

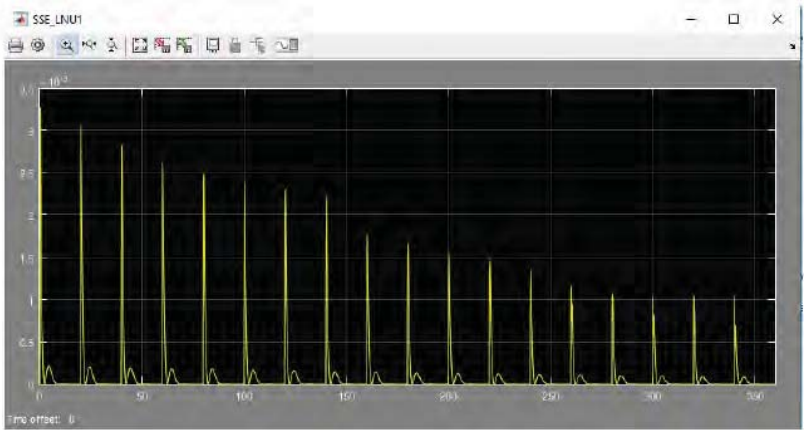

Fig. 12: SSE for LNU with respect to time.

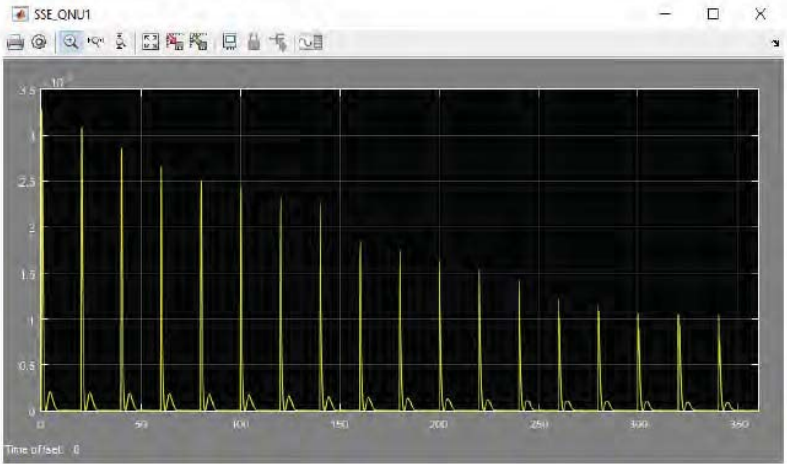

Fig. 13: SSE for QNU with respect to time.

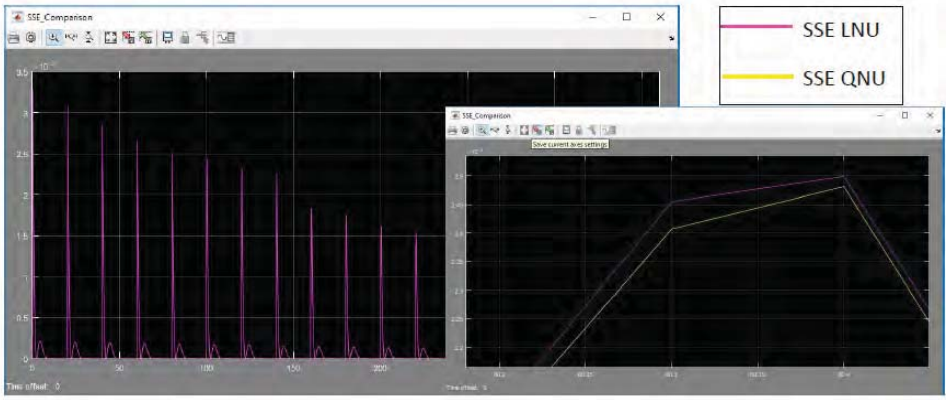

Fig. 14: SSE for QNU with respect to time

7 Conclusion

Following the results of this work, a MATLAB Simulink toolbox for adaptive identification was created and implemented on a theoretical linear system as a plant. To satisfy better understanding for readers, the fundamental gradient descent learning algorithm for LNU and further QNU architectures were recalled. From the produced experimental results, an important property was shown behind the comparison of a linear versus a quadratic neural unit architecture. From the presented results, it is evident that the QNU reduced its incremental SSE quicker than the same parameter settings and learning algorithm of a linear architecture. This highlights the capabilities of HONUs for efficient real time dynamic system identification in comparison to more complex conventional neural network (NN) architectures as such MLPs, which often require longer incremental training for similar order of approximation strength. Following the outcomes of this project, a future goal of this research is to extend the produced toolbox for application to the CTU roller rig, with focus to multiple-input-multiple-output (MIMO) configuration.

References

[1] P. Benes and I. Bukovsky, "Neural network approach to hoist deceleration control," in *2014 International Joint Conference on Neural Networks (IJCNN)*, 2014, pp. 1864–1869.

[2] I. Bukovsky, P. Benes, and M. Slama, "Laboratory Systems Control with Adaptively Tuned Higher Order Neural Units," in *Intelligent Systems in Cybernetics and Automation Theory*, R. Silhavy, R. Senkerik, Z. K. Oplatkova, Z. Prokopova, and P. Silhavy, Eds. Springer International Publishing, 2015, pp. 275–284.

[3] L. Smetana, "Nonlinear Neuro-Controller for Automatic Control Laboratory System," Master's Thesis, Czech Technical University in Prague, Prague, Czech Republic, 2008.

[4] P. M. Benes, I. Bukovsky, M. Cejnek, and J. Kalivoda, "Neural Network Approach to Railway Stand Lateral Skew Control," in *Computer Science & Information Technology (CS& IT)*, Sydney, Australia, 2014, vol. 4, pp. 327–339.

[5] I. Bukovsky and N. Homma, "An Approach to Stable Gradient-Descent Adaptation of Higher Order Neural Units," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 9, pp. 2022–2034, Sep. 2017.

[6] P. Benes, I. Bukovsky, and J. Kalivoda, "Achievements in Neural Network Approach to Railway Stand Lateral Skew Control," presented at the Nové metody a postupy v oblasti přístrojové techniky, automatického řízení a informatiky 2014, Herbertov, Czech Republic, 2014.

Selected article from
Tento dokument byl publikován ve sborníku

Nové metody a postupy v oblasti přístrojové techniky, automatického řízení a informatiky 2018
New Methods and Practices in the Instrumentation, Automatic Control and Informatics 2018
28. 5. – 30. 5. 2018, Příbram - Podlesí

ISBN 978-80-01-06477-1

Web page of the original document:
http://control.fs.cvut.cz/nmp
http://iat.fs.cvut.cz/nmp/2018.pdf

Obsah čísla/individual articles:
http://iat.fs.cvut.cz/nmp/2018/