COMPUTATIONALLY SIMPLE NEURAL NETWORK APPROACH TO DETERMINE PIECEWISE-LINEAR DYNAMICAL MODEL

P. Dolezel[*], *J. Heckenbergerova*[*]

Abstract: The article introduces a new technique for nonlinear system modeling. This approach, in comparison to its alternatives, is straight and computationally undemanding. The article employs the fact that once a nonlinear problem is modeled by a piecewise-linear model, it can be solved by many efficient techniques. Thus, the result of introduced technique provides a set of linear equations. Each of the equations is valid in some region of state space and together, they approximate the whole nonlinear problem. The technique is comprehensively described and its advantages are demonstrated on an example.

Key words: *artificial neural network, modeling, nonlinear systems*

Received: October 20, 2015 **DOI:** 10.14311/NNW.2017.27.020
Revised and accepted: August 17, 2017

1. Introduction

Piecewise-linear functions provide a useful and attractive tool to deal with nonlinear problems. Once a nonlinear problem is approximated by some piecewise-linear function, it is possible to divide it into a set of linear subproblems where each of them can be solved by some efficient algorithm. In addition, the description of the problem using piecewise-linear functions may be worthwhile for its simplicity of implementation and calculation. This idea was originally proposed in [5] and was expanded in [15, 16, 23] or [4].

Moving to systems theory, piecewise-linear systems are systems whose state space is divided into a finite number of regions, where each individual subsystem is linear [20]. Such systems have received a lot of attention for their equivalence to several other classes of the systems [13], but mainly for their usability. They offer a very attractive model structure which is capable of approximating nonlinear systems by switching between various linear submodels. Consequently, they are useful for system analysis using techniques, which were originally proposed for dealing with linear systems. Besides, special tools for a piecewise-linear system analysis were published, too [21].

[*]Petr Dolezel – Corresponding author; Jana Heckenbergerova; University of Pardubice, Studentska 95, Pardubice, Czech Republic, E-mail: petr.dolezel@upce.cz, jana.heckenbergerova@upce.cz

Modeling of the nonlinear systems using piecewise-linear model is generally a nontrivial problem. In a very special case, when the state space partition is known, the problem converts into a classical linear system identification; the submodel parameters can be estimated from the corresponding input-output data. However, the proper challenge is the situation when the state space partition is not known a priori. An extensive number of literature sources deal with identification of piecewise-linear systems and more general piecewise-affine systems. Ferrari et al. propose clustering techniques [10], while others use the bounded error method [2], an algebraic approach [24] or other methods. A nice review of the recent methods is covered in [11].

In [11], the approaches using state-space models as well as input-output models are considered. Dealing with input-output models, the approaches are categorized as optimization-based methods, algebraic methods, clustering based methods and recursive methods. The first three types of the approach are based on offline identification. Hence, input-output data is collected in an separated experiment and subsequently used to fit a model to the obtained data. However, in most cases, the model structure has to be set a priori. Using the fourth category, the model parameters are considered as time varying states and are adaptively tuned based on estimation errors. The issue of model structure, however, remains as well as in the previous approaches.

Thus, following sections introduce a new technique that can efficiently provide a currently valid linear model (i.e. the model which is valid in some defined region of the state space) of the nonlinear system. Basically, it is a two-step procedure. The first step is implemented offline. On the other hand, the second one works online and it is computationally so simple that it can be applied using a basic microprocessor. The first engineering approach related to this idea was presented at the scientific conference [8] as a tool for process control. Then, it was expanded and applied directly to a controller tuning in [9]. In this paper, the generalized and comprehensive description of the idea, including algorithms and tools for appropriate application, is presented. In this form, the contribution distinctively exceeds the limited information introduced in mentioned sources and it can adapted into a wide family of domains.

The paper is organized as follows: in Sec. 2, the main aim of the contribution is introduced. Secs. 3 and 4 bring tools used for the problem solution and Sec. 5 describes the proposed technique for solution of the problem defined in Sec. 2. The contribution is concluded with an illustrative example.

2. Problem formulation

The aim of the paper is to provide a useful tool for system analysis. For the purposes of the paper, let us narrow a system to a deterministic discrete single-input-single-output system as seen in Fig. 1. A discrete system can be thought as a transformation that maps an input sequence $u(k)$ to an output sequence $y(k)$, where k is a discrete time instant. Output of the system can be formally expressed as follows:

$$y(k) = f\left(y(k-1), \cdots, y(k-N), u(k-1), \cdots, u(k-M), k\right), M \leq N, \quad (1)$$

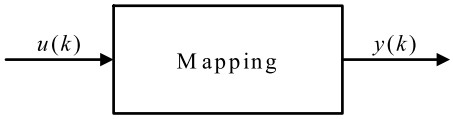

Fig. 1 *Discrete system.*

where $f(\cdot)$ is a nonlinear mapping and N is the order of the system.

As mentioned in the first section, a majority of recent system analysis techniques require a mathematical model of the system to be analyzed. It means that we need a set of equations, which allows us to predict the future behavior of the system. In the following formulas, the predicted value of the system output is denoted as $\hat{y}(k)$.

In linear dynamical system identification, there are available several kinds of models; some of them are stochastic, others are stochastic with exogenous input or even fully deterministic. A widely accepted standard is established in [18]. A general linear model describing deterministic and stochastic influences is obtained by Eq. (2), where all the discrete time series are represented in form of Z-transforms:

$$\hat{Y}(z^{-1}) = G(z^{-1})U(z^{-1}) + H(z^{-1})V(z^{-1}). \tag{2}$$

$U(z^{-1})$ in the equation above is the Z-transform of the input signal $u(k)$ to the system, $\hat{Y}(z^{-1})$ is the Z-transform of the predicted output signal $\hat{y}(k)$ and $V(z^{-1})$ is the transform of white noise. The filter $G(z^{-1})$ is called the input transfer function, since it relates the input to the output. Similarly, $H(z^{-1})$ is called the noise transfer function. Considering the deterministic system (1), stochastic influences in a model (2) can be omitted, assuming that the input sequence $u(k)$ is known, i.e.

$$\hat{Y}(z^{-1}) = G(z^{-1})U(z^{-1}). \tag{3}$$

Dealing with system analysis, it is helpful to replace $G(z^{-1})$ with a ratio of polynomials, as follows:

$$\hat{Y}(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})}U(z^{-1}), \tag{4}$$

where A and B are polynomials of complex variable z^{-1}.

$$A(z^{-1}) = [1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}], \tag{5}$$

$$B(z^{-1}) = [0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_M z^{-M}], \quad M \leq N. \tag{6}$$

The crucial task of system identification is to determine the polynomials $A(z^{-1})$, $B(z^{-1})$, which are then used for system analysis. In addition, if the system is significantly nonlinear, the coefficients of both polynomials shift, depending on the operating point. Therefore, the aim of the article is to introduce a technique of determining the polynomials $A(z^{-1})$, $B(z^{-1})$, which are valid in a defined neighborhood of a current state of the system. The technique is supposed to be efficient enough to be used online and should work as seen in Fig. 2.

The idea depicted above is not original, however the way of determination of the polynomials $A(z^{-1})$, $B(z^{-1})$ is new. It uses a special topology of an artificial neural network and contrary to related techniques, it is computationally simple.

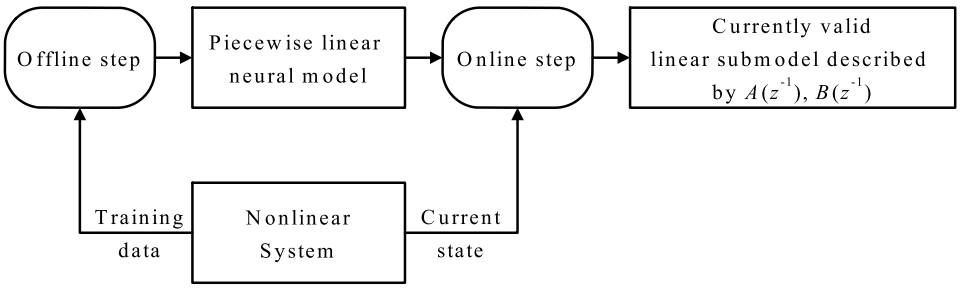

Fig. 2 *The behavior of the technique.*

3. Artificial neural network for universal approximation

In 1989, Hornik proved that a standard multilayer feedforward neural network (FFNN) with one hidden layer is capable of approximating any real measurable function to any desired degree of accuracy [14]. The topology of a FFNN with one hidden layer is depicted in Fig. 3, where the input layer brings external inputs x_1, $x_2, \ldots, x_P$; the hidden layer contains S neurons which process sums of weighted inputs; and an output neuron processes the sum of weighted outputs from hidden neurons. Data flow between an input i and a hidden neuron j is gained by a weight $w^1_{j,i}$. Data flow between the hidden neuron j and the output neuron is gained by a weight $w^2_{1,j}$. Neurons in the hidden layer contain a squashing activation function, while the output neuron contains a non-squashing activation function – see [14] for formal definition. For practical applications, a continuous, bounded and monotonic activation function is used for the neurons in the hidden layer, and a continuous and monotonic activation function is used in the output neuron – examples in [12, 19].

The output of the network in Fig. 3 can be expressed by following equations:

$$y^1_{aj} = \sum_{i=1}^{P} w^1_{j,i} x_i + w^1_j, \tag{7}$$

$$y^1_j = \phi^1\left(y^1_{aj}\right), \tag{8}$$

$$y^2_{a1} = \sum_{j=1}^{S} w^2_{1,j} y^1_j + w^2_1, \tag{9}$$

$$y_{\mathrm{NN}} = \phi^2\left(y^2_{a1}\right). \tag{10}$$

In the equations above, $\phi^1(\cdot)$ means activation functions of hidden neurons and $\phi^2(\cdot)$ means the output neuron activation function. Apparently, the network has to be well trained in order to achieve sufficient approximation qualities. In other words, the network has to learn associations between a specified set of input-output pairs (training set). As many training techniques were presented in many sources of literature, from the simple back-propagation algorithm [22] to some specialized hybrid techniques using evolutionary computation [3], they are not defined here. However, the important note is that analytical derivatives of activation functions

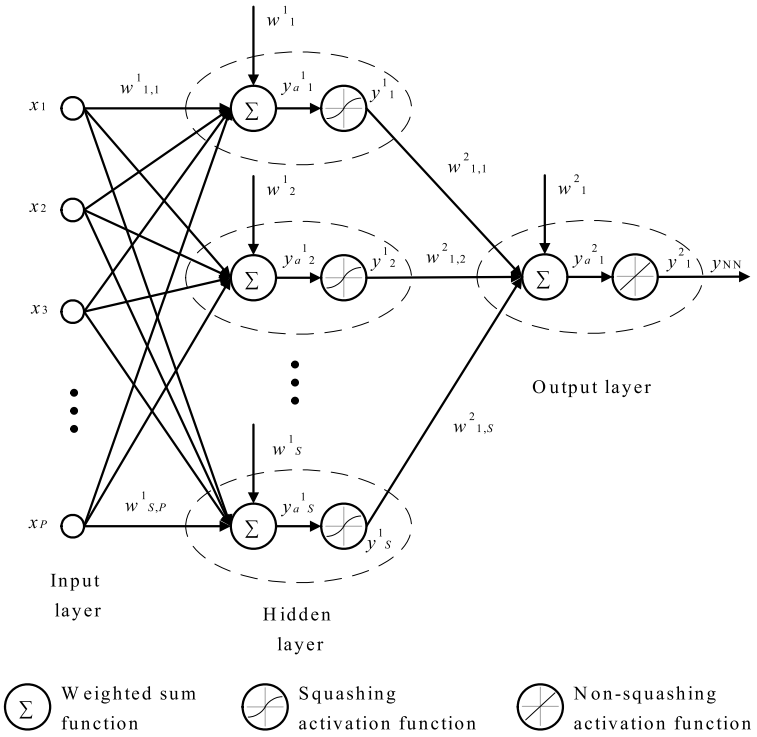

Fig. 3 *Multilayer feedforward neural network with one hidden layer.*

are required for training by any gradient-based technique.

4. System identification by FFNN

System identification is a statistical procedure that leads to a mathematical model of a dynamical system from measured data. System identification using FFNN (whole procedure is defined in [12]) provides dynamical neural models. Output of the FFNN in a neural model can be determined directly by Eqns. (7)–(10). Dynamical neural models are able to model very precisely even highly nonlinear systems because FFNN are universal approximators [1]. See Fig. 4 where a typical chart of a dynamical neural model is shown.

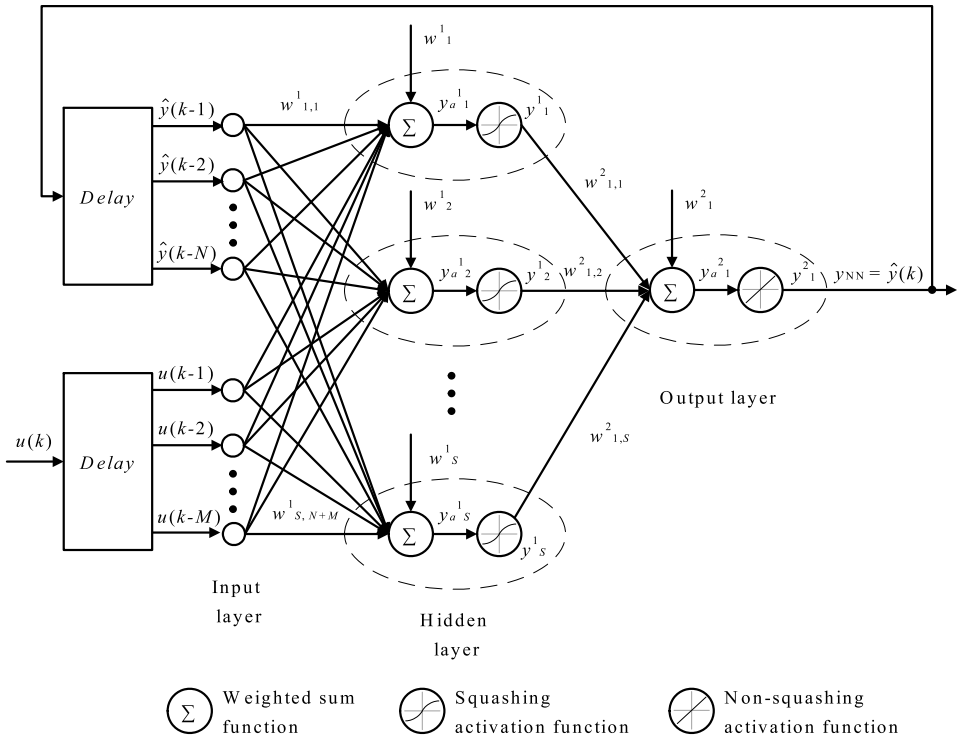

Fig. 4 *Dynamical neural model.*

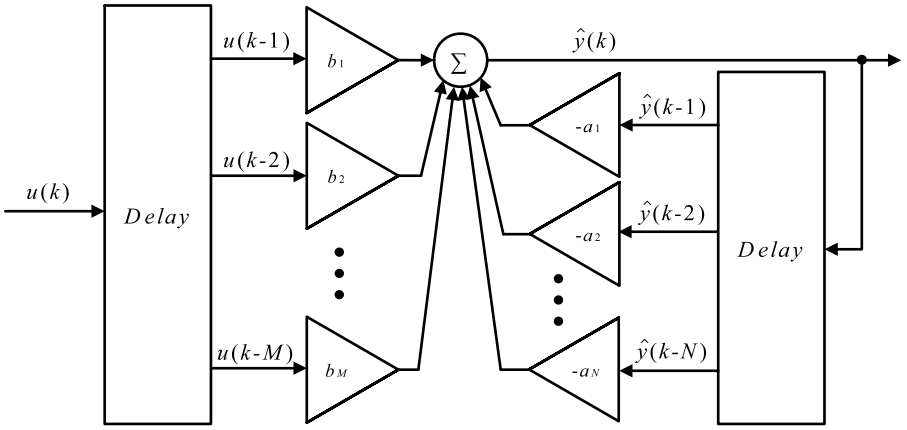

Fig. 5 *Deterministic linear model.*

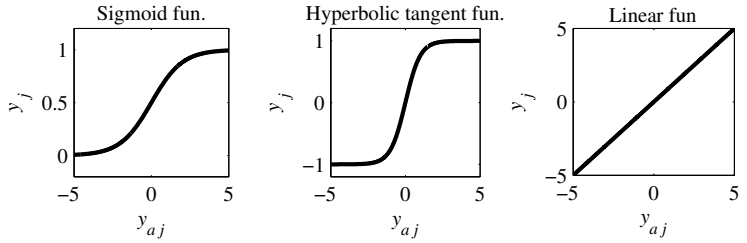

Fig. 6 *Activation functions.*

The issue is that the dynamical neural model is a black-box-like structure and it cannot be directly used for system analysis using classical techniques. On the other hand, a family of linear models with exogenous input (defined by the coefficients of the polynomials $A(z^{-1})$, $B(z^{-1})$ of Eq. (4)) is widely postulated as a suitable model structure for system analysis – see Fig. 5 for an illustration of the deterministic linear models. Thus, the procedure of transforming a dynamical neural model into the model described in Fig. 5 is proposed in the next paragraph.

5. Piecewise-Linear Neural Model

As mentioned in Sec. 3, FFNN has to contain a squashing activation function in the hidden layer and a non-squashing activation function in the output neuron. In addition, these activation functions are expected to be differentiable so that some gradient-based training technique can be used. Actually, in most cases, a hyperbolic-tangent or a sigmoid activation function is used in the hidden layer; and a linear activation function is used in the output neuron – see Fig. 6.

This article offers another approach. It suggests replacing the hyperbolic tangent activation function in the hidden layer with a linear saturated activation function, i.e.

$$y_j = \begin{cases} 1 & \text{for} \quad y_{aj} > 1, \\ y_{aj} & \text{for} \quad -1 \leq y_{aj} \leq 1, \\ -1 & \text{for} \quad y_{aj} < 1. \end{cases} \tag{11}$$

Although a linear saturated activation function is not fully differentiable, FFNN then becomes a piecewise-linear structure. Furthermore, FFNN approximation qualities are expected to stay similar since both functions follow near exact courses – see Fig. 7.

Let us presume an existence of the dynamical neural model which uses FFNN with linear saturated activation functions in hidden neurons and a linear (identical) activation function in the output neuron. Apparently, this model acts as a piecewise-linear model and one linear submodel turns into another when any hidden neuron becomes saturated or becomes not saturated.

Although the output of FFNN used in this dynamical neural model can be evaluated by Eqns. (7)–(10), another way for FFNN output computing is useful. The aim of this alternative approach is apparently to get the formal expression of the evaluation as close to Eq. (4) as possible.

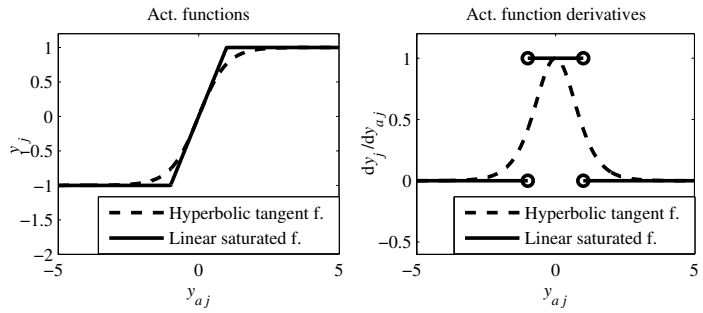

Fig. 7 *Activation functions comparison.*

First, it is necessary to define saturation vector $\mathbf{v}$ of S elements. This vector indicates saturation states of the hidden neurons – see Eq. (12).

$$v_j = \begin{cases} 1 & \text{for} \quad y_j^1 = 1, \\ 0 & \text{for} \quad -1 < y_j^1 < 1, \\ -1 & \text{for} \quad y_j^1 = -1. \end{cases} \quad j = 1, 2, \ldots, S, \tag{12}$$

In other words, the value of v_j indicates one of three different states of neuron j of hidden layer – see Fig. 8.

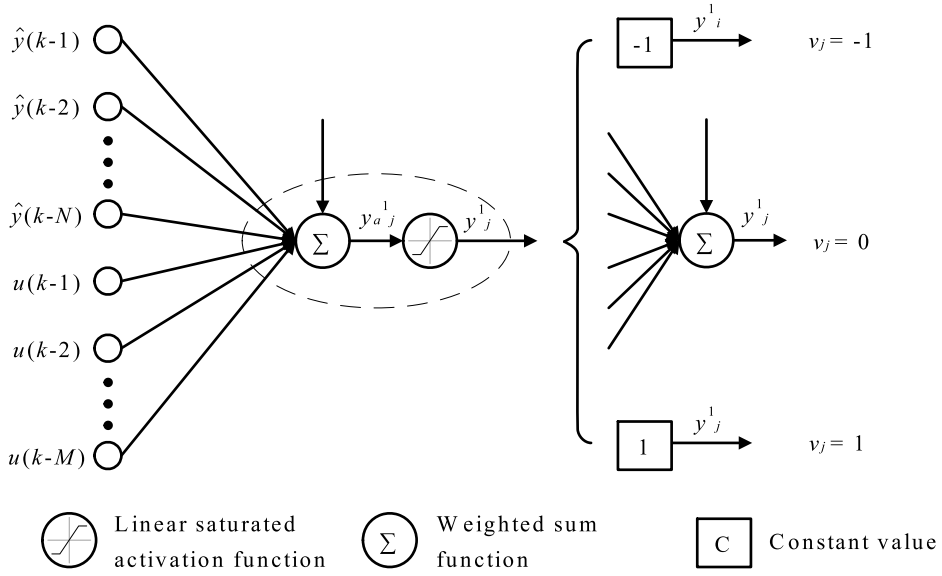

Fig. 8 *Different states of neurons in hidden layer depending on the v_j value.*

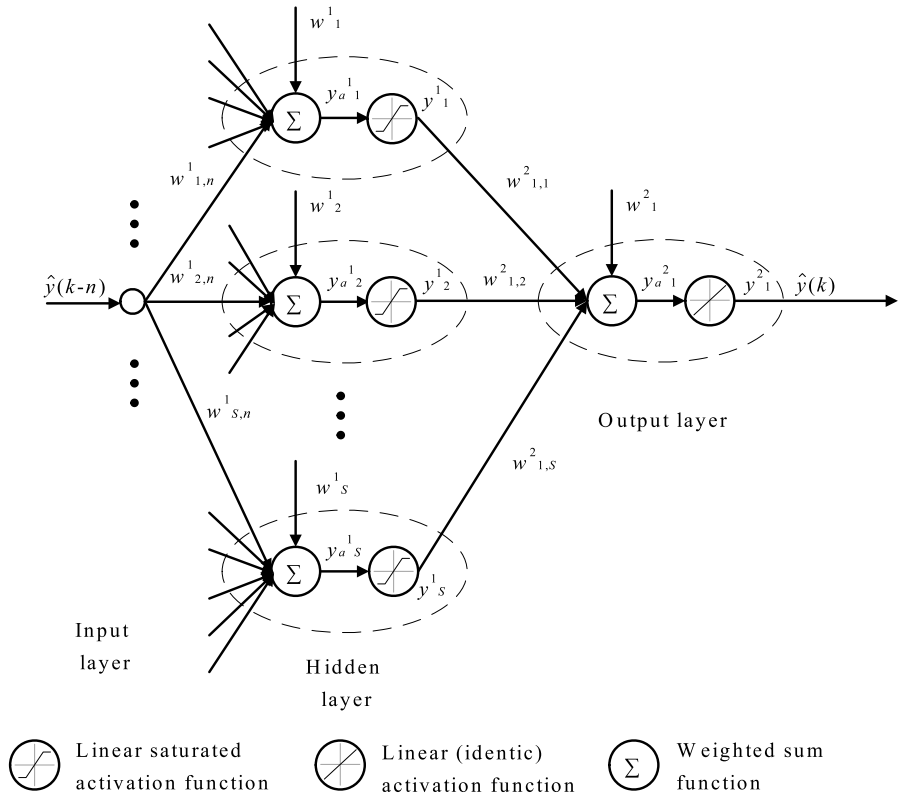

Fig. 9 *Dependency of the output $\hat{y}(k)$ on $\hat{y}(k-n)$.*

Moreover, notice that the term $(1-|v_j|)$ is positive $((1-|v_j|)=1)$ for linear state of the hidden neuron j, and neutral $((1-|v_j|)=0)$ for the saturated state of the hidden neuron.

Let us focus on a region of state space where all elements of vector $\mathbf{v}$ are constant. Inside this region, the output $\hat{y}(k)$ can be expressed as a linear combination of $\hat{y}(k-n), n=1,2,\ldots,N$ and $u(k-m), m=1,2,\ldots,M$. Hence

$$\hat{y}(k) = -\sum_{n=1}^{N} a_n \hat{y}(k-n) + \sum_{m=1}^{M} b_m u(k-m) + c. \tag{13}$$

Fig. 9 can be used in order to determine the coefficients a_n, b_m and c. In this figure, the dependency of the output $\hat{y}(k)$ on generic input $\hat{y}(k-n)$ is illustrated. In a very special case of $v_j = 0, j = 1,2,\ldots,S$ (all linear saturated activation functions are within their linear range), the coefficient a_n could be expressed in the following way

$$a_n = -(w^1_{1,n}w^2_{1,1} + w^1_{2,n}w^2_{1,2} + \cdots + w^1_{S,n}w^2_{1,S}) = -\sum_{j=1}^{S} w^1_{j,n}w^2_{1,j}. \tag{14}$$

Nevertheless, in most cases, some of the linear saturated activation functions are in their saturated states (see Fig. 8). The corresponding neurons of a hidden layer act as sources of a constant signal (either $y_j = -1$ for $v_j = -1$ or $y_j = 1$ for $v_j = 1$). Thus, weights of these neurons should be inactivated from the sum (14). This requirement could be efficiently performed by multiplicating each sum by the term $(1 - |v_j|)$ as shown above. Thus, coefficient a_n can be generally expressed as follows:

$$a_n = -\sum_{j=1}^{S}(1 - |v_j|)w^1_{j,n}w^2_{1,j}. \tag{15}$$

Similarly, a generic coefficient b_m can be expressed in the following way

$$b_m = \sum_{j=1}^{S}(1 - |v_j|)w^1_{j,m+N}w^2_{1,j}. \tag{16}$$

At last, it is necessary to specify the coefficient c as a sum of the bias of output neuron w^2_1, the biases of all the hidden non-saturated neurons w^1_j weighted by corresponding weights of output layer $w^2_{1,j}$ and all the outputs of the hidden saturated neurons (defined by v_j) weighted again by corresponding weights of output layer $w^2_{1,j}$. Thus,

$$c = \sum_{j=1}^{S}\left(w^2_{1,j}v_j + (1 - |v_j|)w^2_{1,j}w^1_j\right) + w^2_1. \tag{17}$$

The difference Eq. (13) defines the FFNN output in some neighborhood of current state (in that neighborhood, where saturation vector $\mathbf{v}$ stays constant). With a change of any element of vector $\mathbf{v}$, the coefficients a_n, b_m and c switch. Nevertheless, the form of Eq. (13) stays the same.

In other words, if the neural model of any nonlinear system in the form of Fig. 4 is designed, then it is simple (applying only Eqns. (15)–(17)) to determine parameters of a linear difference equation, which approximates system behavior in some neighborhood of current state.

The last step is the transformation of the coefficients of Eq. (13) into an expected form (4). Let us define

$$\tilde{u}(k) = u(k) - u_0, \tag{18}$$

where u_0 is constant. Then, Eq. (13) turns into

$$\hat{y}(k) = -\sum_{n=1}^{N} a_n\hat{y}(k-n) + \sum_{m=1}^{M} b_m\tilde{u}(k-m) + c + \sum_{m=1}^{M} b_m u_0. \tag{19}$$

Eq. (19) becomes constant term free, if Eq. (20) is satisfied;

$$u_0 = -\frac{c}{\sum_{m=1}^{M} b_m}. \tag{20}$$

Using Z-transform, Eq. (19) can be now written in the following way:

$$\hat{Y}(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})}\tilde{U}(z^{-1}), \tag{21}$$

where the coefficients of the polynomials $A(z^{-1})$, $B(z^{-1})$ are determined by the Eqns. (15) and (16), respectively. The Eq. (21) corresponds to the model in Fig. 5 with respect to the Eqns. (18) and (20).

The main contribution of the introduced technique is its capability of model transformation into a set of linear submodels, as derived in previous paragraphs (online step according to the Fig. 2). The transformation is computationally very simple; therefore, it can be performed at any time using simple microprocessors.

A number of neurons used in the hidden layer S directly affects the number of linear regions provided by the technique. To be more specific, vector $\mathbf{v}$ can potentially gather up to 3^S states – see Eq. (12). Thus, there are 3^S linear submodels stored in Eq. (21). Transitions between these linear models can be determined by finding the states, where any element of vector $\mathbf{v}$ switches to a different value. In other words, the borders between regions are defined by the following set of equations

$$\begin{aligned} y^1_{aj} &= -1, \\ y^1_{aj} &= 1, \end{aligned} \quad j = 1, 2, \ldots, S. \tag{22}$$

Using the solution of the previous set of equations, the model (21) can be written as follows:

$$\begin{aligned} A^1(z^{-1})\hat{Y}(z^{-1}) &= B^1(z^{-1})\tilde{U}(z^{-1}), \quad \text{if } \mathbf{x} \in \mathbf{X}_1, \\ A^2(z^{-1})\hat{Y}(z^{-1}) &= B^2(z^{-1})\tilde{U}(z^{-1}), \quad \text{if } \mathbf{x} \in \mathbf{X}_2, \\ &\vdots \\ A^R(z^{-1})\hat{Y}(z^{-1}) &= B^R(z^{-1})\tilde{U}(z^{-1}), \quad \text{if } \mathbf{x} \in \mathbf{X}_R. \end{aligned} \tag{23}$$

In the equation above, $A^h(z^{-1}), B^h(z^{-1}), h = 1, 2, \ldots, R$, are the polynomials with constant coefficients. The vector $\mathbf{x}$ defines the current state of the model, $\mathbf{x} = [u(k-1), \cdots, u(k-M), \hat{y}(k-1), \cdots, \hat{y}(k-N)]^{\mathrm{T}}$. Then, $\mathbf{X} = \bigcup_{h \in \{1,2,\cdots,R\}} \mathbf{X}_h$ denotes the state space partition into closed regions. Such a description can be then advantageously used for further processing – the most obvious example is stability analysis, as presented in [17].

The main steps of the presented technique are listed below as a summary of this section.

- To divide a nonlinear system into linear submodels, it is necessary to design a dynamical neural model of the system. Note that neurons in the hidden layer of FFNN should contain a linear saturated activation function and the output neuron contains a linear (identical) activation function. This step is discussed in Sec. 4 and is referred as offline step according to Fig. 2.

- Then, using the current state of the system $\mathbf{x}$, the technique can provide a linear submodel valid in some neighborhood of the current state. See Eqns. (13), (15)–(17), and (21) for a detailed procedure. This step is termed as online step since it is performed continuously or on demand.

- The region of validity of the derived linear submodel, as well as all the other submodels, can be determined using a solution of Eq. (22).

6. Illustrative example

As an example for the main features of the introduced technique, the following issue is dealt with. Consider a Hammerstein system consisting of static nonlinearity in series with second order linear system. The important characteristics of the system are shown in Fig. 10.

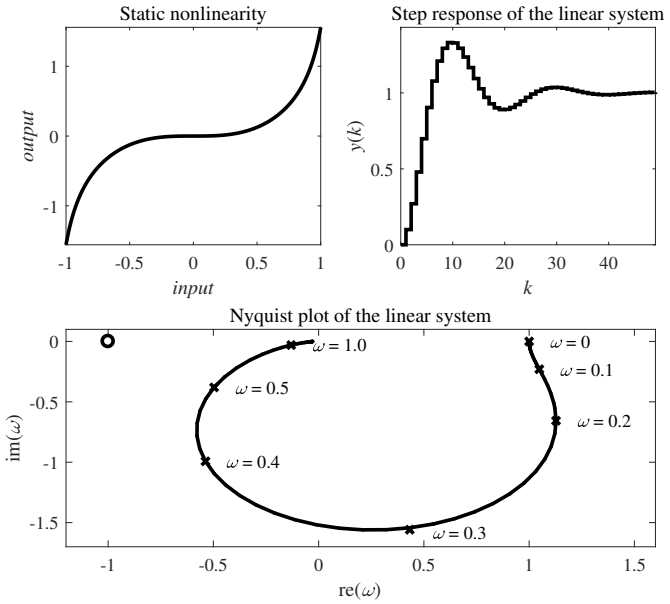

Fig. 10 *Responses of the demonstrative system.*

To divide this system into linear subsystems according to the algorithm described above, it is necessary to design a neural model of the system in the shape of Fig. 4, where neurons in the hidden layer of FFNN contain a linear saturated activation function and the output neuron contains a linear (identical) activation function (offline step according to the Fig. 2). This procedure involves training set acquisition, neural network training and pruning, and neural model validating. As this sequence of processes is illustrated closely in many other publications [12, 19], it is not referred to here in detail. However, main steps are summarized below to keep the experiments repeatable.

Since the point is this section is not to prove accuracy and reliability of the derived model (a legitimacy of this form of modeling has been proven many years ago), only a response to a defined function is performed as a visual check of the resulting neural model.

6.1 Training data

Mostly, the data for training, testing and validation set are gained by measurement of the system response to a defined input signal. In this particular case, the response

to a generic signal is used (see Fig. 11 for a part of the dataset). 5000 samples are collected and divided into a training set (85 %), and validation set. The whole set of collected data is available in [6].

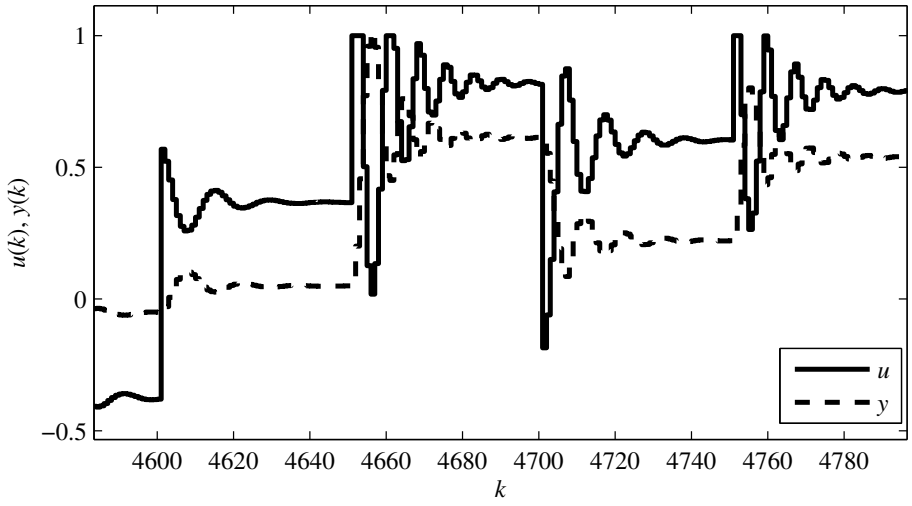

Fig. 11 *A part of the collected data.*

6.2 Order of neural model

The order of the model, i.e. the degrees of the polynomials $A(z^{-1})$, $B(z^{-1})$ is determined experimentally. To be more specific, redundant FFNNs (20 neurons with linear saturated activation functions in hidden layer, 1 output neuron with linear (identical) activation function) are trained using training data ordered into different forms, according to a supposed piecewise-linear neural model. Remaining that $M \leq N$, the following possibilities are considered – see Tab. I.

Number	M	N
1	1	1
2	1	2
3	2	2
4	1	3
5	3	3

Tab. I *Considered polynomial degrees.*

A standard mean square error function computed over the validation set is used as a test criterion, whereas 100 training experiments are performed to achieve higher significancy of the results. According to the previous authors' experience, the Levenberg-Marquardt algorithm is used for training while the Nguyen-Widrow

method is applied for the initialization of the weights. The missing derivatives of the linear saturated activation function for $y_{aj}^1 = \pm 1$ (see Fig. 7) are approximated by the values belonging to the closest neighbors; i.e.

$$\left.\frac{\partial\phi\left(y_{aj}^1\right)}{\partial y_{aj}^1}\right|_{y_{aj}^1=\pm 1} := 1. \tag{24}$$

All the parameters of the experiments are shown below – see Tab. II.

Training algorithm	Levenberg-Marquardt algorithm
Initialization	Nguyen-Widrow method
Maximum epochs	20
Stopping criterion	Maximum epochs reached
Adaptive coefficient μ	0.001
Increment μ	10
Decrement μ	0.1

Tab. II *Parameters of the experiments.*

Resulting values of the mean square error function for various polynomial degrees are illustrated in Fig. 12. For all the box graphs there, the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually.

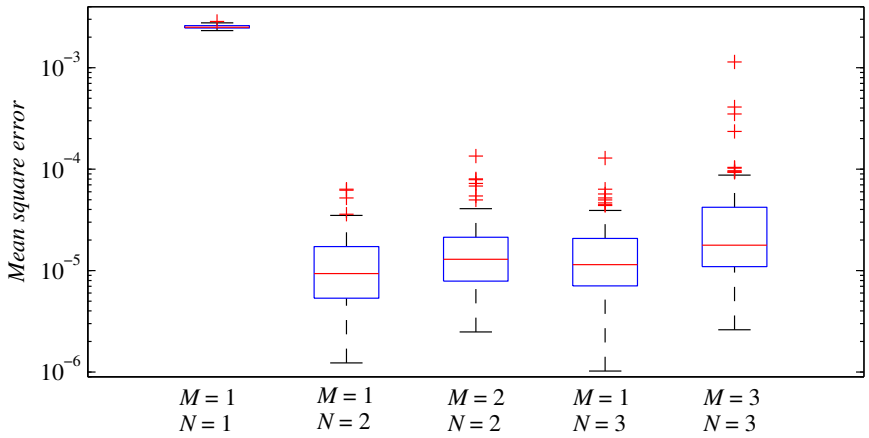

Fig. 12 *Order of the model.*

Considering Fig. 12, there is a significant gap between the first and the second box graph, while the rest of the graphs provide similar statistical quantities. Thus,

the model with a structure

$$\hat{Y}(z^{-1}) = \frac{\left[0 + b_1 z^{-1}\right]}{\left[1 + a_1 z^{-1} + a_2 z^{-2}\right]} \tilde{U}(z^{-1}) \tag{25}$$

is considered as suitable.

6.3 Optimal topology of FFNN used in neural model

Since the order of the model is decided, it is necessary to determine the optimal topology of a piecewise-linear FFNN used in a dynamical neural model and, of course, the proper weight values. Similarly to the previous set of trials, the number of neurons in the hidden layer is set experimentally.

Therefore, a family of training experiments is performed again. Now, the polynomial degrees remain the same ($M = 1$, $N = 2$), but the number of the neurons in the hidden layer varies. In conformity with previous trainings, 100 experiments are performed for each topology. In this case, the resulting network will be considered as suitable for application. Thus, the number of epochs is raised to 100 to ensure enough power to find an appropriate combination of weights. Other training parameters remain the same as in the previous experiment – see Tab. II.

Final values of the mean square error function for various topologies in the form of box graphs are illustrated in Fig. 13.

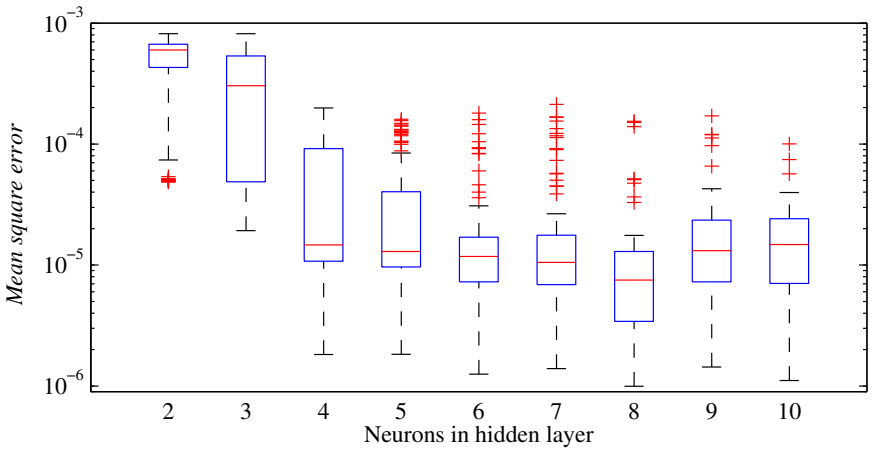

Fig. 13 *Topology of the FFNN used in model.*

According to Fig. 13, the final values of the mean square error functions fall steeply with the rising number of neurons in hidden layer. This situation occurs until point 4, where the values start to stay more or less constant. Thus, the topology of the network is set to 3 inputs, 4 neurons in the hidden layer and 1 output. Clearly, the weights of the resulting network are set according to the most successful training procedure within this topology.

6.4 Neural model analysis

The procedure of transformation into a set of linear submodels is figured here for the derived neural model. The piecewise-linear FFNN prepared in Sec. 6.3 including all the weights values is shown in Fig. 14.

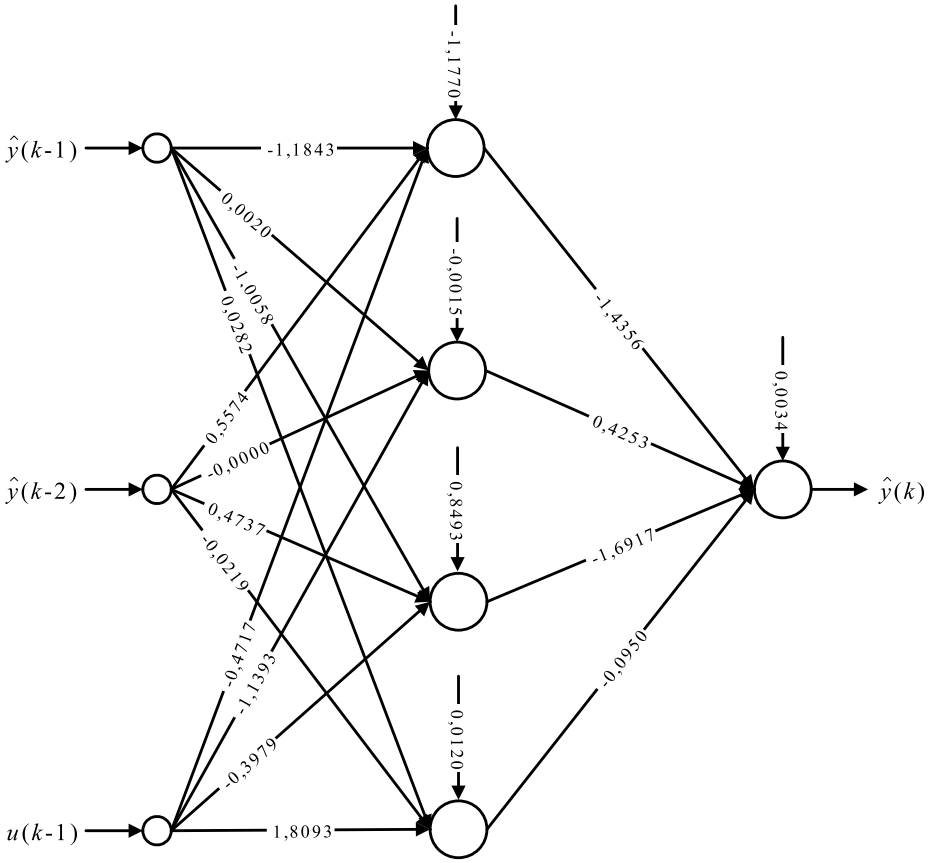

Fig. 14 *Used piecewise-linear neural network.*

Since 4 neurons are used in the hidden layer ($S = 4$) of the used FFNN, vector $\mathbf{v}$ can potentially gather up to 81 states – see Eq. (12). Thus, there are 81 linear submodels stored in the structure shown in Fig. 14. Transitions between these linear models can be determined by finding the states, where any element of vector $\mathbf{v}$ switches to a different value, as shown at the end of Sec. 5.

Using Eq. (7), Eq. (22) transforms into the system of linear equations

$$
\begin{aligned}
&w^1_{1,1}\hat{y}(k-1) + w^1_{1,2}\hat{y}(k-2) + w^1_{1,3}u(k-1) + w^1_1 = -1,\\
&w^1_{1,1}\hat{y}(k-1) + w^1_{1,2}\hat{y}(k-2) + w^1_{1,3}u(k-1) + w^1_1 = 1,\\
&w^1_{2,1}\hat{y}(k-1) + w^1_{2,2}\hat{y}(k-2) + w^1_{2,3}u(k-1) + w^1_2 = -1,\\
&w^1_{2,1}\hat{y}(k-1) + w^1_{2,2}\hat{y}(k-2) + w^1_{2,3}u(k-1) + w^1_2 = 1,\\
&w^1_{3,1}\hat{y}(k-1) + w^1_{3,2}\hat{y}(k-2) + w^1_{3,3}u(k-1) + w^1_3 = -1,\\
&w^1_{3,1}\hat{y}(k-1) + w^1_{3,2}\hat{y}(k-2) + w^1_{3,3}u(k-1) + w^1_3 = 1,\\
&w^1_{4,1}\hat{y}(k-1) + w^1_{4,2}\hat{y}(k-2) + w^1_{4,3}u(k-1) + w^1_4 = -1,\\
&w^1_{4,1}\hat{y}(k-1) + w^1_{4,2}\hat{y}(k-2) + w^1_{4,3}u(k-1) + w^1_4 = 1.
\end{aligned}
\tag{26}
$$

Since a particular state is defined by three variables ($\hat{y}(k-1)$, $\hat{y}(k-2)$, $u(k-1)$) for this system, a solution of each equation above creates a plane dividing the state space into two semispaces. Thus, the whole state space with these boundaries can be visualized by a 3D graph – see Fig. 15. The parameters of the model in the form of Eq. (25) can be computed directly by using the Eqns. (15)–(17).

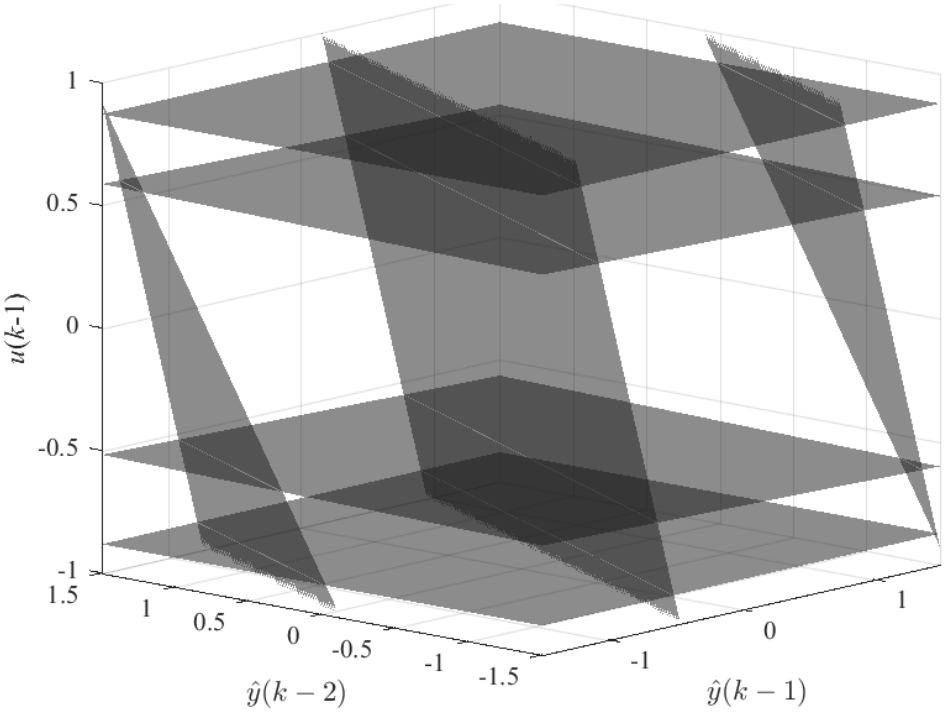

Fig. 15 *State space map with linear regions.*

A response of the original system and its piecewise-linear neural model to a defined signal is performed – see Fig. 16. For a detail of the response, see Fig. 17. Note that the neural model works as an autonomous system, not only a one step predictor (i.e. parallel architecture of the model is applied). In Fig. 16, the regions, in which particular linear models are used, are also marked and numbered. The numbering corresponds with Tab. III. See that only a fragment of 81 computed regions is applied. Many of them lie in the frontier area of state space and they can

be even discarded from the further analysis if their position in state space proves their infeasibility.

Looking at the particular values in Tab. III, it is obvious that parameters a_1 and a_2 remain (almost) the same during the response, while b_1 and c shift with the operation point. This result corresponds with the Hammerstein type of system – a nonlinearity is caused only by the static part of the system – see the characteristics in Fig. 10.

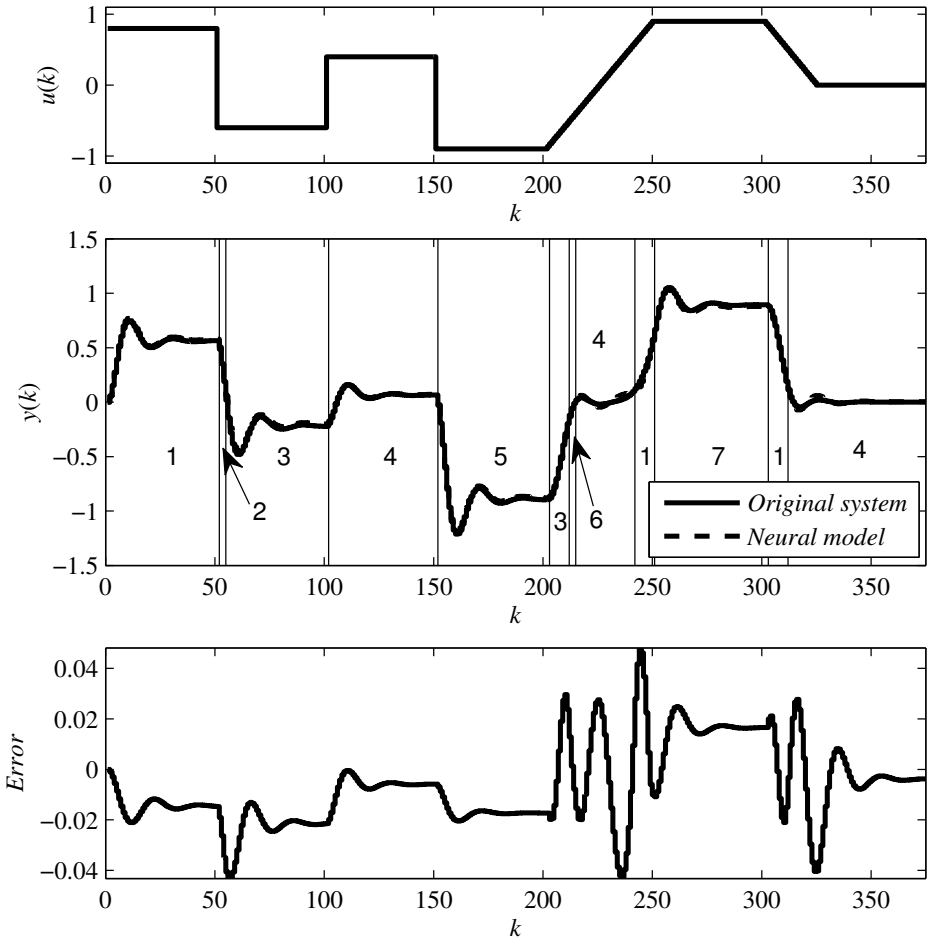

Fig. 16 *Response of original system and its piecewise-linear model (the third graph shows the difference between the output of the original system and its neural model).*

Hence, the technique is able to determine a set of linear models of the nonlinear system, and it is even possible to compute the regions of validity of these linear models.

It brings several advantages in comparison to other techniques which provide a piecewise-linear model. The set of linear submodels is stored in a memory-effective structure and a particular linear submodel can be determined on demand. The

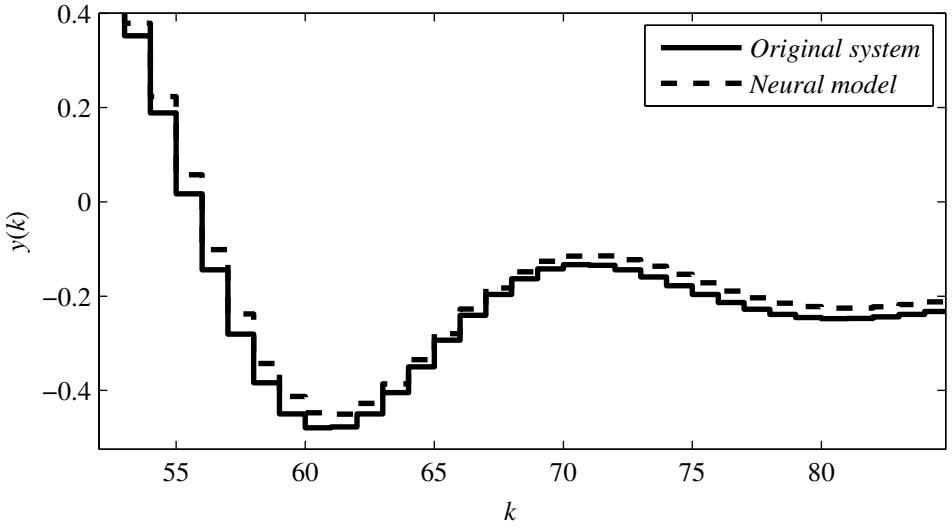

Fig. 17 *Detail of Fig. 16.*

Number	a_1	a_2	b_1	c
1	−1.7024	0.8017	0.1885	−0.0935
2	−1.7024	0.8017	0.1885	0.0966
3	−1.7010	0.8006	0.1926	0.0959
4	−1.6997	0.7996	0.0165	0.0004
5	−1.7002	0.8003	0.6772	0.5218
6	−1.6984	0.7985	0.0206	−0.0003
7	−1.7015	0.8014	0.6731	−0.5182

Tab. III *Used linear submodels – see Eqns. (25), (21), and (20).*

number of linear submodels is related to the neural network complexity, which is designed to reach a defined precision of a neural model. This feature advantageously avoids the necessity of the user to set the number of linear submodels. And last but not least, the process of a neural model design (which is the crucial step of the technique) is known, reliable and well examined procedure. From the computational complexity point of view, the linearization is performed using Eqns. (15)–(17), which are basically summations of several elements with some additions and multiplications. For an example, in this particular case, each step of linearization costs less than 10^{-4} sec using a mainstream personal computer. Recently, the technique has also been applied in industrial environment. The computational times using robust (but slow) industrial equipment have been acceptable, too. See [7] for more information.

7. Conclusions

The aim of this article is to introduce a novel technique for system identification using a special topology of a feedforward neural network. The technique is described in Secs. 3–5 and the functionality of the technique is demonstrated in Sec. 6.

Since only a feedforward neural network with linear or piecewise-linear elements is applied for nonlinear system identification, it is possible to transform the resulting model into a linear submodel valid in the neighborhood of a current system state. Thus, the resulting model effectively stores a finite number of linear models (each valid in some region of state space) and these linear models can be applied locally. Moreover, the illustrative example shows that the presented technique is computationally inexpensive and therefore real time applicable.

The technique can be used in two ways. It is possible to determine a piecewise-linear model of the system, and that structure can be used online for continuous linearization of the system (which is a very effective procedure). On the other hand, the structure of the piecewise-linear model can be divided offline into a set of linear models and comprehensively analyzed.

Nevertheless, both possibilities bring a decent tool to possible nonlinear system control, stability investigation, prediction, fault diagnosis, etc.

References

[1] BALARA D., TIMKO J., ZILKOVA J. Application of neural network model for parameters identification of non-linear dynamic system. *Neural Network World*. 2013, 23(2), pp. 103–116, doi: 10.14311/NNW.2013.23.008.

[2] BEMPORAD A., GARULLI A., PAOLETTI S., VICINO A. Data classification and parameter estimation for the identification of piecewise affine models. In: *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 1, 2004, pp. 20–25, doi: 10.1109/cdc.2004.1428600.

[3] BLANCO A., DELGADO M., PEGALAJAR M.C. A real-coded genetic algorithm for training recurrent neural networks. *Neural Networks*. 2001, 14(1), pp. 93–105, doi: 10.1016/s0893-6080(00)00081-2.

[4] BREIMAN L. Hinging hyperplanes for regression, classification, and function approximation. *Information Theory, IEEE Transactions on*. 1993, 39(3), pp. 999–1013, doi: 10.1109/18.256506.

[5] CHUA L., KANG S.M. Section-wise piecewise-linear functions: Canonical representation, properties, and applications. *Proceedings of the IEEE*. 1977, 65(6), pp. 915–929, doi: 10.1109/proc.1977.10589.

[6] DOLEZEL P. Data for nonlinear system identification, 2016. URL: https://www.researchgate.net/publication/312474730_Data_for_nonlinear_system_identification.

[7] DOLEZEL P., GAGO L. Piecewise linear neural network for process control in industrial environment. In: *2016 17th International Carpathian Control Conference (ICCC)*, 2016, pp. 161–165. doi: 10.1109/CarpathianCC.2016.7501086.

[8] DOLEZEL P., HAVLICEK L., DUSEK F. Elevation control of helicopter model using piecewise-linear neural network. In: *International Conference on Applied Electronics*, 2011, pp. 103–106.

[9] DOLEZEL P., TAUFER I. Piecewise-linear artificial neural networks for pid controller tuning. *Acta Montanistica Slovaca*, 2012, 17(3), pp. 224–233.

[10] FERRARI-TRECATE G., MUSELLI M., LIBERATI D., MORARI M. A clustering technique for the identification of piecewise affine systems. *Automatica*. 2003, doi: 10.1016/s0005-1098(02)00224-8.

[11] GARULLI A., PAOLETTI S., VICINO A. A survey on switched and piecewise affine system identification. *IFAC Proceedings Volumes*. 2012, 45(16), pp. 344–355, URL: http://www.sciencedirect.com/science/article/pii/S1474667015379751, doi: http://dx.doi.org/10.3182/20120711-3-BE-2027.00332.

[12] HAYKIN S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999. ISBN: 0132733501.

[13] HEEMELS W.P.M.H., SCHUTTER B.D., BEMPORAD A. On the equivalence of classes of hybrid dynamical models. In: *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, vol. 1, 2001, pp. 364–369, doi: 10.1109/.2001.980127.

[14] HORNIK K., STINCHCOMBE M., WHITE H. Multilayer feedforward networks are universal approximators. *Neural Networks*. 1989, 2(5), pp. 359–366, doi: 10.1016/0893-6080(89)90020-8.

[15] HUANG X., XU J., WANG S. Nonlinear system identification with continuous piecewise linear neural network. *Neurocomputing*. 2012, 77(1), pp. 167–177, doi: 10.1016/j.neucom.2011.09.001.

[16] LAI C.Y., XIANG C., LEE T.H. Identification and control of nonlinear systems via piecewise affine approximation. In: *Decision and Control (CDC), 2010 49th IEEE Conference on* (Dec 2010), pp. 6395–6402. doi: 10.1109/cdc.2010.5717032.

[17] LIU X., ZHAO X. Stability analysis of discrete-time switched systems: a switched homogeneous lyapunov function method. *International Journal of Control*. 2016, 89(2), pp. 297–305, doi: 10.1080/00207179.2015.1075254.

[18] LJUNG L. *System identification – Theory for the User*. Prentice-Hall, 1999.

[19] NGUYEN H., PRASAD N., WALKER C. *A First Course in Fuzzy and Neural Control*. Chapman and Hall/CRC. ISBN: 1584882441.

[20] PAOLETTI S., JULOSKI A., FERRARI-TRECATE G., VIDAL R. Identification of hybrid systems a tutorial. *European Journal of Control*. 2007, 13(2-3), pp. 242–260, doi: 10.3166/ejc.13.242-260.

[21] PWLTool, a Matlab Toolbox for Piecewise Linear System. Tech. rep., Department of Automatic Control, Lund Institute of Technology (LTH), 1999.

[22] RUMELHART D.E., HINTON G.E., WILLIAMS R.J. Learning representations by back-propagating errors. *Nature*. 1986, doi: 10.1038/323533a0.

[23] SONTAG E.D. Nonlinear regulation: The piecewise linear approach. *Automatic Control, IEEE Transactions on*. 1981, 26(2), pp. 346–358, doi: 10.1109/tac.1981.1102596.

[24] VIDAL R., SOATTO S., MA Y., SASTRY S. An algebraic geometric approach to the identification of a class of linear hybrid systems. In: *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on* (Dec. 2003), vol. 1, pp. 167–172, doi: 10.1109/cdc.2003.1272554.