Int. J. Appl. Math. Comput. Sci., 2010, Vol. 20, No. 1, 23–34
DOI: 10.2478/v10006-010-0002-x

LOCAL STABILITY CONDITIONS FOR DISCRETE–TIME CASCADE LOCALLY RECURRENT NEURAL NETWORKS

KRZYSZTOF PATAN

Institute of Control and Computation Engineering
University of Zielona Góra, ul. Podgórna 50, 65–246 Zielona Góra, Poland
e-mail: k.patan@issi.uz.zgora.pl

The paper deals with a specific kind of discrete-time recurrent neural network designed with dynamic neuron models. Dynamics are reproduced within each single neuron, hence the network considered is a locally recurrent globally feedforward. A crucial problem with neural networks of the dynamic type is stability as well as stabilization in learning problems. The paper formulates local stability conditions for the analysed class of neural networks using Lyapunov's first method. Moreover, a stabilization problem is defined and solved as a constrained optimization task. In order to tackle this problem, a gradient projection method is adopted. The efficiency and usefulness of the proposed approach are justified by using a number of experiments.

Keywords: locally recurrent neural network, stability, stabilization, learning, constrained optimization.

Notation

$\boldsymbol{u}$	input vector
$\boldsymbol{y}$	output vector
$\boldsymbol{x}^1$	state vector of 1-st layer
$\boldsymbol{x}^2$	state vector of 2-nd layer
$\boldsymbol{v}^1$	state vector of autonomous system 1-st layer
$\boldsymbol{v}^2$	state vector of autonomous system 2-nd layer
$\boldsymbol{A}_i$	state matrix of i-th neuron
$\boldsymbol{A}^j$	state matrix of j-th layer
a^j_{1i}, a^j_{2i}	first and second feedback filter parameters of i-th neuron in j-th layer
$\boldsymbol{W}^j$	weight matrix of j-th layer
$\boldsymbol{W}^u$	weight matrix between input and 2-nd layer
$\boldsymbol{B}^1$	feedforward filter parameters matrix of 1-st layer
b^1_{1i}, b^1_{2i}	first and second feedforward filter parameters of i-th neuron in 1-st layer
$\boldsymbol{D}^1$	transfer matrix of 1-st layer
$\boldsymbol{g}^1_1$	vector of biases of 1-st layer
$\boldsymbol{G}^1_2$	slope parameters matrix of 1-st layer
g^1_{2i}	slope parameter of i-th neuron in 1-th layer
$\boldsymbol{\sigma}(\cdot)$	vector-valued activation function
$\boldsymbol{C}$	output matrix
v_1, v_2	neuron number in 1-st and 2-nd layer
r	filter order

1. Introduction

In the last decade, a growing interest in locally recurrent networks has been observed. This class of neural networks, due to their interesting properties, has been successfully applied to solve problems from different scientific and engineering areas. Cannas and co-workers (2001) applied a locally recurrent network to train the attractors of Chua's circuit, as a paradigm for studying chaos. The modelling of continuous polymerisation and neutralisation processes is reported in (Zhang *et al*., 1998). In turn, a three-layer locally recurrent neural network was successfully applied to the control of nonlinear systems in (Gupta and Rao, 1993). In the framework of fault diagnosis, the literature reports many applications, e.g., an observer based fault detection and isolation system of a three-tank laboratory system (Marcu *et al*., 1999), or model based fault diagnosis of sensor and actuator faults in a sugar evaporator (Patan and Parisini, 2005). Tsoi and Back (1994) compared and applied different architectures of locally recurrent networks to the prediction of speech utterance. Finally, Campolucci and Piazza (2000) elaborated an intristic stability control method for a locally recurrent network designed for signal processing.

Stability plays an important role in both control theory and system identification. Furthermore, the stability issue is of crucial importance in relation to training

algorithms adjusting the parameters of neural networks. If the predictor is unstable for certain choices of neural model parameters, serious numerical problems can occur during training. Stability criteria should be universal, applicable to as broad a class of systems as possible and, at the same time, computationally efficient. The majority of well-known approaches are based on Lyapunov's method (Gupta *et al*., 2003; Ensari and Arik, 2005; Cao *et al*., 2006; Forti *et al*., 2005).

Stability analysis for locally recurrent networks with only one hidden layer is given in (Patan, 2007). Unfortunately, approximation abilities of such networks are limited (Patan, 2008a). Therefore, there is a need to derive stability criteria for more complex locally recurrent networks. Recently, global stability of the locally recurrent network with two hidden layers based on Lyapunov's second method was investigated in (Patan, 2008c). Unfortunately, theorems based on Lyapunov's second method formulate sufficient conditions for global asymptotical stability of the system, and they cannot be used as a starting point to determine constraints on the network parameters. This paper presents an approach, based on the first method of Lyapunov, which allows us to elaborate a training procedure with constraints on the network parameters. Thus, the training process can guarantee the stability of the neural model.

The paper is organized as follows: In Section 2, the locally recurrent network and its representations in the state-space are described. Stability analysis of the neural network considered as well as the stabilization procedure are given in Section 3. Illustrative examples of stable training of the examined neural network are provided in Section 4. Section 5 includes conlusions and final remarks.

2. Locally recurent networks

A biological neural cell not only contains a nonlinear mapping operation on a weighted sum of its inputs, but it also has some dynamic properties such as state feedbacks, time delays hysteresis or limit cycles. In order to cope with such dynamic behaviour, a special kind of neuron model has been proposed (Gori *et al*., 1989; Back and Tsoi, 1991; Fasconi *et al*., 1992; Gupta and Rao, 1993). Such neuron models constitute a basic building block for designing a complex dynamic neural network.

The dynamic neuron unit systematized by Gupta and co-workers in (Gupta *et al*., 2003) as the basic element of neural networks of the dynamic type receives not only external inputs but also state feedback signals from itself and other neurons in the network. The synaptic links in this model contain a self-recurrent connection representing a weighted feedback signal of its state and lateral connections which constitute state feedback from other neurons of the network. The dynamic neuron unit is connected to other $(n-1)$ models of the same type forming a neural network. Neural networks composed of dynamic neuron units have a recurrent structure with lateral links between neurons.

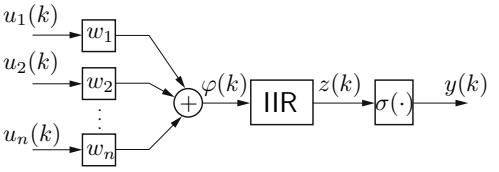

Fig. 1. Neuron architecture with the IIR filter.

A different approach providing dynamically driven neural networks is used in the so-called Locally Recurrent Globally Feed-forward (LRGF) networks (Tsoi and Back, 1994; Patan, 2008b). LRGF networks have an architecture that is somewhere in-between a feedforward and a globally recurrent one. The topology of such a kind of neural network is analogous to the multi-layered feedforward one, and the dynamics are reproduced by the so-called dynamic neuron models. Based on the well-known McCulloch-Pitts neuron model, various dynamic neuron models can be designed. In general, differences between these depend on the localization of internal feedbacks.

One of the possible solutions is to use linear dynamics in the structure of the neuron. The dynamics are introduced to the neuron in such a way that neuron activation depends on its internal states. This is done by introducing an Infinite Impulse Response (IIR) filter into the neuron structure. In this way, the neuron reproduces its own past inputs and activations using two signals: the input $u_i(k)$, for $i = 1, 2, \ldots, n$, and the output $y(k)$. The weights perform a similar role as in static feedforward networks. The weights, together with the activation function, are responsible for approximation properties of the model. Then this calculated sum $\varphi(k)$ is passed to the IIR filter. Here, the filters under consideration are linear dynamic systems of different orders, viz. the first or the second order. The filter consists of feedback and feedforward paths weighted by suitable weights. Finally, based on the signal $z(k)$ received from the filter, the neuron generates its output using a nonlinear activation function $\sigma(\cdot)$.

One of the main advantages of locally recurrent networks is that their structure is similar to that of static feedforward ones. The dynamic neurons replace the standard static neurons. This network structure does not have any global feedbacks, which complicate the architecture of the network and the training algorithm. In general, the dynamic network can include one or more hiddent layers containing dynamic neuron models. The number of hidden layers directly influences approximation abilities of the network model. In the work of Patan (2008a), it was proved that a locally recurent network consisting of two hidden layers of neurons with IIR filters is able to approx-

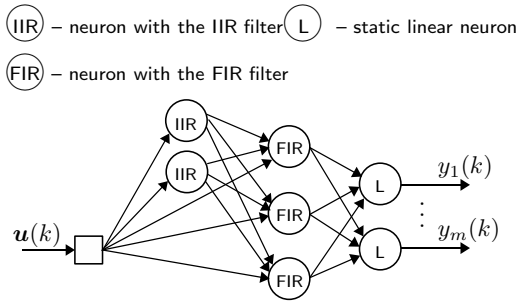

Fig. 2. Cascade structure of the locally recurrent neural network.

imate a state-space trajectory produced by any Lipschitz continuous function with arbitrary accuracy. Moreover, in that paper a new, less complex, neural structure was proposed. In the following section, details about a cascade neural network and its representation in the state-space are portrayed.

2.1. Cascade network. Let us consider a discrete-time neural network with n inputs and m outputs. The cascade locally recurrent network is composed of two processing layers consisting of v_1 and v_2 neurons, respectively. Neurons of the second layer receive excitation not only from the neurons of the previous layer but also from external inputs (Fig. 2) (Patan *et al*., 2008). In this case, the second layer of the network is not a hidden one, contrary to the original structure of locally recurrent networks (Patan and Parisini, 2005).

The first layer includes neurons with IIR filters while the second one consists of neurons with Finite Impulse Response (FIR) filters. Each neuron consists of a filter of order r. The state-space representation block schemes of both kind of neurons are presented in Figs. 3 and 4.

Neuron with the IIR filter. The states of the i-th neuron can be described by the following state equation:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}_i\boldsymbol{x}(k) + \boldsymbol{W}_i\boldsymbol{u}(k), \tag{1}$$

where $\boldsymbol{x}(k) \in \mathbb{R}^r$ is the state vector, $\boldsymbol{W}_i = \mathbf{1}\boldsymbol{w}_i^T$ is the weight matrix ($\boldsymbol{w}_i \in \mathbb{R}^n$, $\mathbf{1} \in \mathbb{R}^r$ is the vector with one

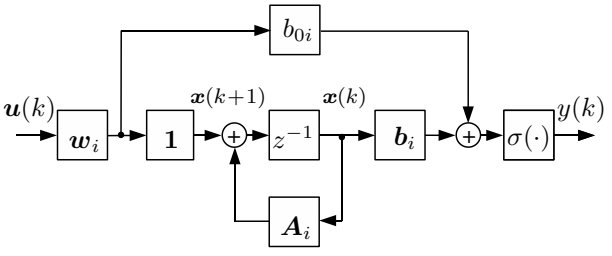

Fig. 3. State-space form of the i-th neuron with the IIR filter.

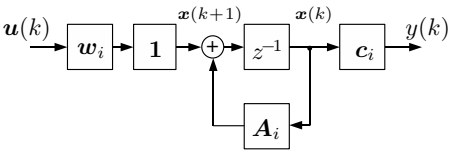

Fig. 4. State-space form of the i-th neuron with the FIR filter.

in the first place and zeros elsewhere), $\boldsymbol{u}(k) \in \mathbb{R}^n$ is the input vector, n is the number of inputs, and the state matrix $\boldsymbol{A}_i$ has the form

$$\boldsymbol{A}_i = \begin{bmatrix} -a_{1i} & -a_{2i} & \dots & -a_{r-1i} & -a_{ri} \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}. \tag{2}$$

Finally, the neuron output is described by

$$y(k) = \sigma\big(g_{2i}(\boldsymbol{b}_i^T\boldsymbol{x}(k) + \boldsymbol{d}_i^T\boldsymbol{u}(k) - g_{1i})\big), \tag{3}$$

where $\sigma(\cdot)$ is a nonlinear activation function, $\boldsymbol{b}_i = [b_{1i}, \dots, b_{ri}]^T$ is the vector of feedforward filter parameters, $\boldsymbol{d}_i = [b_{0i}w_{1i}, \dots, b_{0i}w_{ni}]^T$, g_{1i} and g_{2i} are the bias and slope of the activation function, respectively.

Neuron with the FIR filter. The states of the i-th neuron with the FIR filter are represented by (1), which is the same as for the neuron with IIR filter. The difference is in the representation of the observation equation. The neuron output is described as follows:

$$y(k) = \boldsymbol{c}_i^T\boldsymbol{x}(k), \tag{4}$$

where $\boldsymbol{c}_i \in \mathbb{R}^r$ is the output vector.

2.2. State-space representation of the cascade locally recurrent network. The state of the cascade network is represented as follows:

$$\boldsymbol{x}^1(k+1) = \boldsymbol{A}^1\boldsymbol{x}^1(k) + \boldsymbol{W}^1\boldsymbol{u}(k), \tag{5a}$$

$$\boldsymbol{x}^2(k+1) = \boldsymbol{A}^2\boldsymbol{x}^2(k) + \boldsymbol{W}^2\boldsymbol{\sigma}\big(\boldsymbol{G}_2^1(\boldsymbol{B}^1\boldsymbol{x}^1(k) + \boldsymbol{D}^1\boldsymbol{u}(k) - \boldsymbol{g}_1^1)\big) + \boldsymbol{W}^u\boldsymbol{u}(k), \tag{5b}$$

where $\boldsymbol{x}^1(k) \in \mathbb{R}^{N_1}$ ($N_1 = v_1 \times r$) represents the states of the first layer, and $\boldsymbol{x}^2(k) \in \mathbb{R}^{N_2}$ ($N_2 = v_2 \times r$) represents the states of the second layer, $\boldsymbol{A}^1 \in \mathbb{R}^{N_1 \times N_1}$ and $\boldsymbol{A}^1 \in \mathbb{R}^{N_2 \times N_2}$ are the block diagonal state matrices of the first and second layers, respectively, $\boldsymbol{W}^1 \in \mathbb{R}^{N_1 \times n}$ is the input weight matrix, $\boldsymbol{W}^2 \in \mathbb{R}^{N_2 \times v_1}$ is the weight matrix between the first and second layers, $\boldsymbol{W}^u \in \mathbb{R}^{N_2 \times n}$ is the weight matrix between the input and the second layer, $\boldsymbol{B}^1 \in \mathbb{R}^{v_1 \times N_1}$ is the block diagonal matrix of feedforward filter parameters of the first layer, $\boldsymbol{D}^1 \in \mathbb{R}^{v_1 \times n}$ is

the transfer matrix, $\boldsymbol{g}_1^1 \in \mathbb{R}^{v_1}$ denotes the vector of biases of the first layer, $\boldsymbol{G}_2^1 \in \mathbb{R}^{v_1 \times v_1}$ is the diagonal matrix of slope parameters of the first layer, and $\boldsymbol{\sigma} : \mathbb{R}^{v_1} \to \mathbb{R}^{v_1}$ is the nonlinear vector-valued function.

The presented cascade neural network possesses prety good approximation abbilities. In the work of Patan (2008b), it was proved that the cascade network (5) with a suitably large number of neurons with IIR filters in the first layer and a suitably large number of neurons with FIR filters in the second layer is able to approximate a state-space trajectory produced by any Lipschitz continuous function with arbitrary accuracy.

The network structure (5) is not a strict feedforward one as it has a cascade structure. The introduction of an additional weight matrix $\boldsymbol{W}^u$ renders it possible to obtain a system equivalent to the classical locally recurrent network with two hidden layers (Patan and Parisini, 2005), but the main advantage of this representation is that the whole state vector is available from the neurons of the second layer of the network. This fact is of crucial importance, taking into account the training of the neural network. If the output $\boldsymbol{y}(k)$ is

$$\boldsymbol{y}(k) = \boldsymbol{x}^2(k), \tag{6}$$

then weight matrices can be determined using a training process, which minimizes the error between the network output and measurable states of the process. Usually, in engineering practice, not all process states are directly available (measurable). In such cases, the dimension of the output vector is rather lower than that of the state vector, and the network output can be produced in the following way:

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}^2(k). \tag{7}$$

In such cases, the cascade neural network contains an additional layer of static linear neurons playing the role of the output layer (Fig. 2).

2.3. Model transformation. In order to derive stability conditions for the network considered, some necessary model transformations are required. Consider the neural model represented by the state equation (5). Let $\boldsymbol{\Psi} = \boldsymbol{G}_2^1\boldsymbol{B}^1$ and $\boldsymbol{s}_1(k) = \boldsymbol{G}_2^1\boldsymbol{D}^1\boldsymbol{u}(k) - \boldsymbol{G}_2^1\boldsymbol{g}_1^1$, where $\boldsymbol{s}_1(k)$ can be treated as a threshold or a fixed input. Then (5b) takes the form

$$\begin{aligned}\boldsymbol{x}^2(k+1) = \boldsymbol{A}^2\boldsymbol{x}^2(k) + \boldsymbol{W}^2\boldsymbol{\sigma}\big(\boldsymbol{\Psi}\boldsymbol{x}^1(k) + \boldsymbol{s}_1(k)\big) \\ + \boldsymbol{W}^u\boldsymbol{u}(k).\end{aligned} \tag{8}$$

Using the linear transformation $\boldsymbol{v}^1(k) = \boldsymbol{\Psi}\boldsymbol{x}^1(k) + \boldsymbol{s}_1(k)$ and $\boldsymbol{v}^2(k) = \boldsymbol{x}^2(k)$, one obtains an equivalent system:

$$\begin{cases}\boldsymbol{v}^1(k+1) = \boldsymbol{\Psi}\boldsymbol{A}^1\boldsymbol{\Psi}^-\boldsymbol{v}^1 - \boldsymbol{\Psi}\boldsymbol{A}^1\boldsymbol{\Psi}^-\boldsymbol{s}_1 + \boldsymbol{s}_2(k), \\ \boldsymbol{v}^2(k+1) = \boldsymbol{A}^2\boldsymbol{v}^2(k) + \boldsymbol{W}^2\boldsymbol{\sigma}\big(\boldsymbol{v}^1(k)\big) + \boldsymbol{s}_3(k),\end{cases} \tag{9}$$

where $\boldsymbol{\Psi}^-$ is a pseudoinverse of the matrix $\boldsymbol{\Psi}$, $\boldsymbol{s}_2(k) = \boldsymbol{\Psi}\boldsymbol{W}^1\boldsymbol{u}(k) + \boldsymbol{s}_1(k)$ and $\boldsymbol{s}_3(k) = \boldsymbol{W}^u\boldsymbol{u}(k)$ are thresholds or fixed inputs.

Let $\boldsymbol{v}^* = [\boldsymbol{v}^{1*}\ \boldsymbol{v}^{2*}]^T$ be an equilibrium point of (9). Introducing an equivalent coordinate transformation $\boldsymbol{z}(k) = \boldsymbol{v}(k) - \boldsymbol{v}^*(k)$, the system (9) can be transformed to the following form:

$$\begin{cases}\boldsymbol{z}^1(k+1) = \boldsymbol{\Psi}\boldsymbol{A}^1\boldsymbol{\Psi}^-\boldsymbol{z}^1(k), \\ \boldsymbol{z}^2(k+1) = \boldsymbol{A}^2\boldsymbol{z}^2(k) + \boldsymbol{W}^2\boldsymbol{f}(\boldsymbol{z}^1(k)),\end{cases} \tag{10}$$

where $\boldsymbol{f}(\boldsymbol{z}^1(k)) = \boldsymbol{\sigma}(\boldsymbol{z}^1(k) + \boldsymbol{v}^{1*}(k)) - \boldsymbol{\sigma}(\boldsymbol{v}^{1*}(k))$. Substituting $\boldsymbol{z}(k) = [\boldsymbol{z}^1(k)\ \boldsymbol{z}^2(k)]^T$, one finally obtains

$$\boldsymbol{z}(k+1) = \mathcal{A}\boldsymbol{z}(k) + \mathcal{W}\boldsymbol{f}(\boldsymbol{z}(k)), \tag{11}$$

where

$$\mathcal{A} = \begin{bmatrix}\boldsymbol{\Psi}\boldsymbol{A}^1\boldsymbol{\Psi}^- & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}^2\end{bmatrix}, \quad \mathcal{W} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{W}^2 & \boldsymbol{0}\end{bmatrix}. \tag{12}$$

3. Stability analysis

Theorems based on Lyapunov's second method formulate sufficient conditions for global asymptotical stability of a system. In many cases, however, there is a need to determine neccessary conditions. In such cases, Lyapunov's first method can be used. Moreover, stability criteria developed using the second method of Lyapunov cannot be used as a starting point to determime constraints on the network parameters. Thus, the optimization problem with constraints cannot be determined. This section presents an approach, based on the first method of Lyapunov, which allows us to elaborate a training procedure with constraints on the network parameters. Thus, the training process can guarantee the stability of the neural model.

Lemma 1. (Lyapunov's first method) *Let $\boldsymbol{x}^* = 0$ be an equilibrium point of the system*

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k)), \tag{13}$$

where $\boldsymbol{f} : D \to \mathbb{R}^n$ is a continuously differentiable function and D is a neighbourhood of the origin. Define the Jacobian of (13) in the neigbourhood of the equilibrium point $\boldsymbol{x}^* = 0$ as

$$\boldsymbol{J} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=0}. \tag{14}$$

Then

1. *The origin is locally asymptotically stable if all the eigenvalues of $\boldsymbol{J}$ are inside the unit circle in the complex plane.*
2. *The origin is unstable if one or more of the eigenvalues of $\boldsymbol{J}$ are outside the unit circle in the complex plane.*

Theorem 1. *The neural system (11) composed of neurons with first order filters ($r = 1$) is locally asymptotically stable if the following conditions are satisfied:*

1. $|a_{1i}^j| < 1, \quad \forall i = 1, \ldots, v_1, \quad \forall j = 1, 2,$
2. $b_{1i}^1 \neq 0, \quad \forall i = 1, \ldots, v_1,$

where a_{1i}^j represents the only element of the state matrix of the i-th neuron in the j-th layer, and b_{1i}^1 represents the only element of the feedforward filter parameters vector of the i-th neuron in the first layer.

Proof. The Jacobian of (11) is given by

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{\Psi}\boldsymbol{A}^1\boldsymbol{\Psi}^- & \boldsymbol{0} \\ \boldsymbol{W}^2\boldsymbol{f}'(0) & \boldsymbol{A}^2. \end{bmatrix}. \tag{15}$$

The characteristic equation has the form

$$\det(\boldsymbol{J} - \lambda\boldsymbol{I}) = 0. \tag{16}$$

The Jacobian is a block matrix, and then the determinant of $\boldsymbol{J} - \lambda\boldsymbol{I}$ is given by

$$\begin{aligned} \det(\boldsymbol{J} - \lambda\boldsymbol{I}) &= \det(\boldsymbol{\Psi}\boldsymbol{A}^1\boldsymbol{\Psi}^- - \lambda\boldsymbol{I})\det(\boldsymbol{A}^2 - \lambda\boldsymbol{I}) \\ &\quad - \det(\boldsymbol{W}^2\boldsymbol{\sigma}'(0))\cdot 0 \\ &= \det(\boldsymbol{\Psi}\boldsymbol{A}^1\boldsymbol{\Psi}^- - \lambda\boldsymbol{I})\det(\boldsymbol{A}^2 - \lambda\boldsymbol{I}). \end{aligned} \tag{17}$$

Finally, the characteristic equation takes the form

$$\det(\boldsymbol{\Psi}\boldsymbol{A}^1\boldsymbol{\Psi}^- - \lambda\boldsymbol{I})\det(\boldsymbol{A}^2 - \lambda\boldsymbol{I}) = 0, \tag{18}$$

and the system is stable if all eigenvalues of both matrices $\boldsymbol{\Psi}\boldsymbol{A}^1\boldsymbol{\Psi}^-$ and $\boldsymbol{A}^2$ are located in the unit circle. In our case,

$$\begin{aligned} \boldsymbol{A}^1 &= \text{diag}(-a_{11}^1, \ldots, -a_{1v_1}^1), \\ \boldsymbol{A}^2 &= \text{diag}(-a_{11}^2, \ldots, -a_{1v_2}^2), \\ \boldsymbol{B}^1 &= \text{diag}(b_{11}^1, \ldots, b_{1v_1}^1). \end{aligned}$$

If Condition 2 is satisfied, then

$$\boldsymbol{\Psi} = \text{diag}(g_{21}^1 b_{11}^1, \ldots, g_{2v_1}^1 b_{1v_1}^1).$$

In this trivial case, a pseudoinverse of $\boldsymbol{\Psi}$ is given as follows:

$$\boldsymbol{\Psi}^- = \begin{bmatrix} \dfrac{1}{g_{21}^1 b_{11}^1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \dfrac{1}{g_{2v_1}^1 b_{1v_1}^1} \end{bmatrix} \tag{19}$$

and, finally, $\boldsymbol{\Psi}\boldsymbol{A}^1\boldsymbol{\Psi}^- = \boldsymbol{A}^1$. Then the system is stable if all eigenvalues of $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$ are located in the unit circle. Taking into account the reasoning presented in (Patan, 2007; 2008a), one knows that all eigenvalues of a block diagonal matrix are located in the unit circle if the eigenvalues of each matrix on the diagonal are located in the unit circle. According to this, $|a_{1i}^1| < 1, \quad \forall i = 1, \ldots, v_1$ and $|a_{1i}^2| < 1, \quad \forall i = 1, \ldots, v_2$, which completes the proof. ■

Theorem 2. *The neural system (11) composed of neurons with second order filters ($r = 2$) is locally asymptotically stable if the following conditions are satisfied:*

1. *For each entry of $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$, the following set of inequalities is satisfied:*

$$\begin{cases} 1 - a_1 + a_2 > 0, \\ 1 + a_1 + a_2 > 0, \\ 1 - a_2 > 0, \end{cases} \tag{20}$$

2. $(b_{1i}^1)^2 + (b_{2i}^1)^2 \neq 0, \quad \forall i = 1, \ldots, v_1,$
3. $|b_{1i}^1| < |b_{2i}^1|, \quad \forall i = 1, \ldots, v_1,$

where b_{1i}^1 and b_{2i}^1 represent elements of the feedforward filter parameters vector of the i-th neuron in the first layer.

Proof. From the proof of Theorem 1 one knows that the system (11) is stable if the eigenvalues of $\boldsymbol{\Psi}\boldsymbol{A}^1\boldsymbol{\Psi}^-$ and $\boldsymbol{A}^2$ are located in the unit circle. Let us consider the eigenvalues of $\boldsymbol{A}^2$ first. According to the reasoning presented in (Patan, 2007), one knows that the eigenvalues of $\boldsymbol{A}^2$ are stable if, for each entry on the diagonal a set of inequalities, (20) holds. Next, take into account $\boldsymbol{\Psi}\boldsymbol{A}^1\boldsymbol{\Psi}^-$. This is a block diagonal matrix with the entries $\boldsymbol{\Psi}_i\boldsymbol{A}_i^1\boldsymbol{\Psi}_i^-, i = 1, \ldots, v_1$, where $\boldsymbol{\Psi}_i = g_{2i}^1\boldsymbol{b}_i^1$. Using Singular Value Decomposition (SVD), it is easy to verify that

$$\boldsymbol{\Psi}_i^- = \frac{(\boldsymbol{b}_i^1)^T}{g_{2i}^1\|\boldsymbol{b}_i^1\|_2^2}, \tag{21}$$

where $\|\boldsymbol{x}\|_2$ is the Euclidean norm of the vector $\boldsymbol{x}$. Using (21), $\boldsymbol{\Psi}_i\boldsymbol{A}_i^1\boldsymbol{\Psi}_i^-$ can be represented as

$$\boldsymbol{\Psi}_i\boldsymbol{A}_i^1\boldsymbol{\Psi}_i^- = \frac{-a_{1i}^1(b_{1i}^1)^2 + b_{1i}^1 b_{2i}^1(1 - a_{2i}^1)}{(b_{1i}^1)^2 + (b_{2i}^1)^2}. \tag{22}$$

In order to obtain a stable system, the condition

$$\left|\frac{-a_{1i}^1(b_{1i}^1)^2 + b_{1i}^1 b_{2i}^1(1 - a_{2i}^1)}{(b_{1i}^1)^2 + (b_{2i}^1)^2}\right| < 1, \quad \forall i = 1, \ldots, v_1, \tag{23}$$

should be satisfied. To clarify the presentation in the following deliberations, the index i is omitted. Let us rewrite (23) as follows:

$$\begin{aligned} -(b_1^1)^2 - (b_2^1)^2 - b_1^1 b_2^1 &< f(a_1^1, a_2^1) \\ &< (b_1^1)^2 + (b_2^1)^2 - b_1^1 b_2^1, \end{aligned} \tag{24}$$

where $f(a_1^1, a_2^1) = -a_1^1(b_1^1)^2 - a_2^1 b_1^1 b_2^1$. To complete the proof, it is necessary to show that

$$\max f(a_1^1, a_2^1) < (b_1^1)^2 + (b_2^1)^2 - b_1^1 b_2^1$$

and

$$\min f(a_1^1, a_2^1) > -(b_1^1)^2 - (b_2^1)^2 - b_1^1 b_2^1.$$

Therefore, it is required to solve two optimization problems:

$$\begin{aligned} \max \quad & f(a_1^1, a_2^1) \\ \text{s.t.} \quad & 1 - a_1^1 + a_2^1 \geqslant 0, \\ & 1 + a_1^1 + a_2^1 \geqslant 0, \\ & 1 - a_2^1 \geqslant 0 \end{aligned} \tag{25a}$$

and

$$\begin{aligned} \min \quad & f(a_1^1, a_2^1), \\ \text{s.t.} \quad & 1 - a_1^1 + a_2^1 \geqslant 0, \qquad 1 + a_1^1 + a_2^1 \geqslant 0. \\ & 1 - a_2^1 \geqslant 0 \end{aligned} \tag{25b}$$

The slope of the cost function $f(a_1^1, a_2^1)$ is given by

$$\alpha = -\frac{(b_1^1)^2}{b_1^1 b_2^1}. \tag{26}$$

Taking into account the shape of the feasible region, one can consider the following cases:

Case 1. $b_1^1 b_2^1 > 0$ and $|-\alpha| > 1$. Then $(b_1^1)^2 > b_1^1 b_2^1$. The course of the cost function is presented in Fig. 5. The maximum is located at the point $P_1 = (-2, 1)$ and the minimum at the point $P_2(2, 1)$.

Case 2. $b_1^1 b_2^1 > 0$ and $|\alpha| < 1$. Then $(b_1^1)^2 < b_1^1 b_2^1$. The course of the cost function is presented in Fig. 6. The maximum is located at the point $P_3 = (0, 1)$ and the minimum at the point $P_2 = (2, 1)$. There is another posibility, when $b_1^1 b_2^1 > 0$ and $(b_1^1)^2 = b_1^1 b_2^1$ ($|\alpha| = 1$), but in this case $|b_1^1| = |b_2^1|$ and Condition 3 is not satisfied.

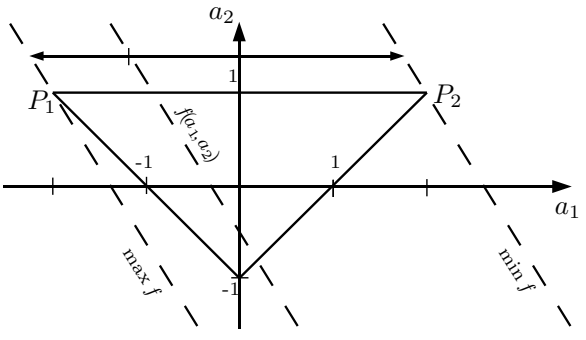

Fig. 5. Graphical solution of the problems (25)—Case 1.

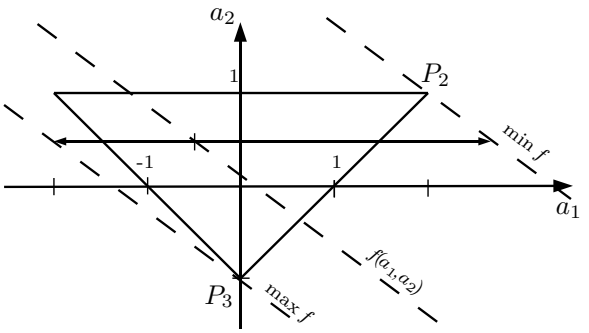

Fig. 6. Graphical solution of the problems (25)—Case 2.

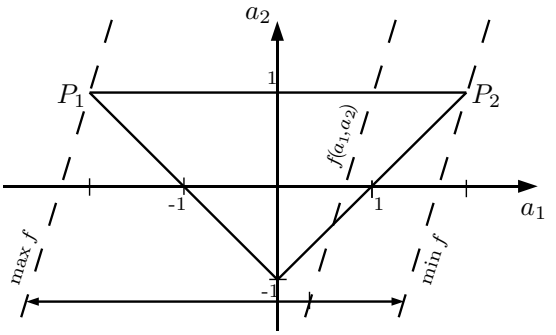

Fig. 7. Graphical solution of the problems (25)—Case 3.

Case 3. $b_1^1 b_2^1 < 0$ and $|\alpha| > 1$. Then $(b_1^1)^2 > -b_1^1 b_2^1$. The course of the cost function is presented in Fig. 7. The maximum is located at the point $P_1 = (-2, 1)$ and the minimum at the point $P_2 = (2, 1)$.

Case 4. $b_1 b_2 < 0$ and $|\alpha| < 1$. Then $(b_1^1)^2 < -b_1^1 b_2^1$. The course of the cost function is presented in Fig. 8. The maximum is located at the point $P_1 = (-2, 1)$ and the minimum at the point $P_3 = (0, -1)$. There is another posibility, when $b_1^1 b_2^1 < 0$ and $(b_1^1)^2 = -b_1^1 b_2^1$ ($|\alpha| = 1$), but in this case $|b_1^1| = |b_2^1|$ and Condition 3 is not satisfied.

According to (24), one should check the following:

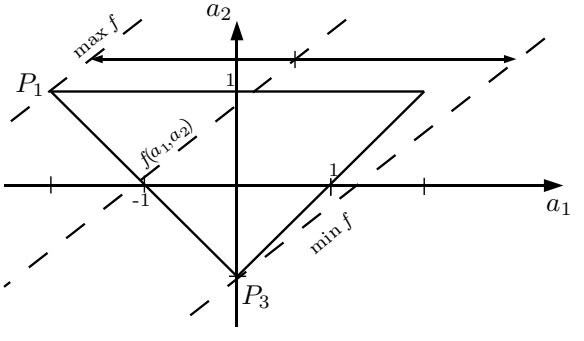

Fig. 8. Graphical solution of the problems (25)—Case 4.

1. $f(-2,1) < (b_1^1)^2 + (b_2^1)^2 - b_1^1 b_2^1$: In this case

$$2(b_1^1)^2 - b_1^1 b_2^1 < (b_1^1)^2 + (b_2^1)^2 - b_1^1 b_2^1,$$

$$(b_1^1)^2 < (b_2^1)^2,$$

and Condition 3 is satisfied;

2. $f(0,-1) < (b_1^1)^2 + (b_2^1)^2 - b_1^1 b_2^1$: One obtains

$$b_1^1 b_2^1 < (b_1^1)^2 + (b_2^1)^2 - b_1^1 b_2^1,$$

$$0 < (b_1^1 - b_2^1)^2,$$

which is true for any b_1 and b_2;

3. $f(2,1) > -(b_1^1)^2 - (b_2^1)^2 - b_1^1 b_2^1$: In this case

$$-2(b_1^1)^2 - b_1^1 b_2^1 > -(b_1^1)^2 - (b_2^1)^2 - b_1^1 b_2^1,$$

$$(b_1^1)^2 < (b_2^1)^2,$$

and Condition 3 is satisfied;

4. $f(0,-1) > -(b_1^1)^2 - (b_2^1)^2 - b_1^1 b_2^1$: In this case

$$b_1^1 b_2^1 > -(b_1^1)^2 - (b_2^1)^2 - b_1^1 b_2^1,$$

$$0 > -(b_1^1 + b_2^1)^2,$$

which is true for any b_1 and b_2.

The problems (25) were solved for constraints in the form of a compact set $\bar{\mathbb{A}}$, but Condition 1 defines an open set of constraints $\mathbb{A}$. Therefore, the operations of maximum and minimum over the compact set can be replaced by the operations of supremum and infimum over the open set as follows:

$$(b_1^1)^2 + (b_2^1)^2 - b_1^1 b_2^1 > \max_{\bar{\mathbb{A}}} f(a_1^1, a_2^1) = \sup_{\mathbb{A}} f(a_1^1, a_2^1), \tag{27}$$

and

$$-(b_1^1)^2 - (b_2^1)^2 - b_1^1 b_2^1 < \min_{\bar{\mathbb{A}}} f(a_1^1, a_2^1) = \inf_{\mathbb{A}} f(a_1^1, a_2^1), \tag{28}$$

which completes the proof. ■

Remark 1. Contrary to the global asymptotical stability theorems based on the second method of Lyapunov (Patan, 2008b; 2008c), Theorems 1 and 2 formulate necessary as well as sufficient conditions for local asymptotical stability of a neural network and are able to judge between the stability and instability of a neural model. Furthermore, based on the conditions formulated by them, a constrained training procedure can be derived, which guarantees the stability of the neural network. An example of such a training procedure for a neural network consisting of second order filters is presented in Table 1.

4. Experiments

This section presents illustrative examples regarding stable training of the cascade neural network. All experiments were carried out using real process data acquired from the laboratory system AMIRA DR300.

Algorithm 1 Assuring the feasibility of the matrix $\boldsymbol{B}^1$.

Step 0. Initiation
Choose the initial network parameters, set ε to a small value, e.g. $\varepsilon = 10^{-5}$

Step 1. Parameters update
Update the network parameters using a training algorithm

Step 2. Assure the feasibility of the matrices $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$, e.g., using gradient projection or minimum distance projection, proposed in (Patan, 2007)

Step 3. Assure the feasibility of the matrix $\boldsymbol{B}^1$ using the following procedure:

Require: $v_1, \varepsilon > 0$ ▷ e.g., $\varepsilon = 10^{-5}$

for $i = 1$ to v_1 **do**
  if $|b_{1i}^1| > |b_{2i}^1|$ **then**
    if $b_1^1 i > 0$ **then**
      if $b_{2i}^1 > 0$ **then**
        $b_{2i}^1 := b_{1i}^1 + \varepsilon$
      else
        $b_{2i}^1 := -b_{1i}^1 - \varepsilon$
      end if
    else
      if $b_{2i}^1 > 0$ **then**
        $b_{2i}^1 := -b_{1i}^1 + \varepsilon$
      else
        $b_{2i}^1 := b_{1i}^1 - \varepsilon$
      end if
    end if
  end if
end for

Step 4. Termination criteria
if (termination criterion satisfied) **then** STOP **else** go to Step 1

4.1. System description. The system shown in Fig. 9 is used to control the rotational speed of a DC motor with a changing load. The laboratory object considered con-

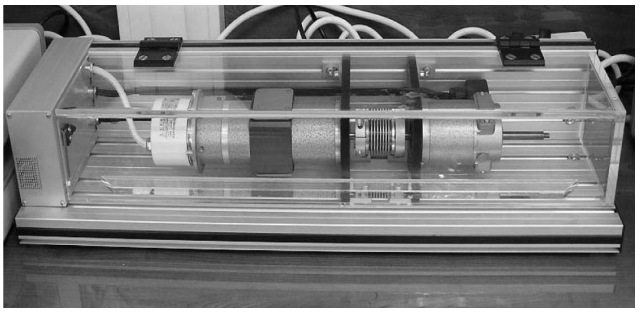

Fig. 9. Laboratory system with a DC motor.

sists of five main elements: a DC motor M1, a DC motor M2, two digital increamental encoders and a clutch K. The input signal of the engine M1 is an armature current and the output one is the angular velocity. The available sensors for the output are an analog tachometer on an optical sensor, which generates impulses that correspond to the rotations of the engine and a digital incremental encoder. The shaft of the motor M1 is connected with the identical motor M2 by the clutch K. The second motor M2 operates in the generator mode and its input signal is an armature current. The available measuremets of the plant are as follows:

- motor current I_m—the motor current of the DC motor M1,
- generator current I_g—the motor current of the DC motor M2,
- tachometer signal T;

and control signals:

- motor control signal C_m—the input of the motor M1,
- generator control signal C_g—the input of the motor M2.

The separately excited DC motor is governed by two differential equations. The classical description of the electrical subsystem is given by the equation

$$u(t) = Ri(t) + L\frac{\mathrm{d}i(t)}{\mathrm{d}t} + e(t), \tag{29}$$

where $u(t)$ is the motor armature voltage, R is the armature coil resistance, $i(t)$ is the motor armature current, L is the motor coil inductance, and $e(t)$ is the induced electromotive force. The counter electromotive force is proportional to the angular velocity of the motor:

$$e(t) = K_e\omega(t), \tag{30}$$

where K_e stands for the motor voltage constant and $\omega(t)$ is the angular velocity of the motor. In turn, the mechanical subsystem can be derived from a torque balance:

$$J\frac{\mathrm{d}\omega(t)}{\mathrm{d}t} = T_m(t) - B_m\omega(t) - T_l - T_f(\omega(t)), \tag{31}$$

where J is the motor moment of inertia, T_m is the motor torque, B_m is the viscous friction torque coefficient, T_l is the load torque, and $T_f(\omega(t))$ is the friction torque.

The motor torque $T_m(t)$ is proportional to the armature current:

$$T_m(t) = K_m i(t), \tag{32}$$

where K_m stands for the motor torque constant. The friction torque can be considered as a function of the angular velocity and it is assumed to be the sum of the Stribeck, Coulumb and viscous components. The viscous friction torque opposes motion and it is proportional to the angular velocity. The Coulomb friction torque is constant at any angular velocity. The Stribeck friction is a nonlinear component occuring at low angular velocities.

Although the model (29)–(32) has a direct relation to the motor physical parameters, the true relation between them is nonlinear. There are many nonlinear factors in the motor, e.g., the nonlinearity of the magnetization characteristic of the material, the effect of material reaction, the effect caused by an eddy current in the magnet, residual magnetism, the commutator characteristic, mechanical frictions (Xiang-Qun and Zhang, 2000). These factors are not shown in the model (29)–(32). Summarizing, the DC motor is a nonlinear dynamic process, and nonlinear modelling should be employed to model it suitably.

The motor described works in closed-loop control with the PI controller. It is assumed that the load of the motor is equal to zero. The objective of system control is to keep the rotational speed at the constant value equal to 2000. Additionally, it is assumed that the reference value is corrupted by additive white noise.

4.2. Motor modelling. A separately excited DC motor was modelled by using the dynamic neural network (5). The model of the motor was selected as follows:

$$T = f(C_m). \tag{33}$$

The following input signal was used in the experiments:

$$\begin{aligned} C_m(k) =& 3\sin(2\pi 1.7k) + 3\sin(2\pi 1.1k - \pi/7) \\ &+ 3\sin(2\pi 0.3k + \pi/3). \end{aligned} \tag{34}$$

The input signal (34) is persistantly exciting of order 6. Using (34), a learning set containig 1000 samples was formed. The objective of the experiment was to compare a standard training procedure with the constrained one. The investigated models were tested using two sets: data generated in the open loop control consisting of 1000 samples (data set T_o) different from the training one, and data generated in the closed loop control consisting of 3500 samples (data set T_c). Results are reported in the following sections.

4.3. Experiment 1. The neural network model (5) had the following structure: one input, three IIR neurons with second order filters and sigmoidal activation functions, four FIR neurons with second order filters and linear activation functions, and one linear output neuron. In all cases considered, the training process was carried out for 100 steps using the Adaptive Random Search (ARS) algorithm with the initial variance $v_0 = 0.1$. Firstly, ARS with gradient projection was employed and the experiment was repeated using the simple ARS algorithm. Figure 10 shows

Table 1. Feedforward filter parameters—neurons with IIR filters.

i	1	2	3
b_{1i}^1	-0.0699	0.8337	-0.8672
b_{2i}^1	0.5863	1.4868	1.2921

the location of the feedback filter parameters in the stability triangle for every single neuron as well as eigenvalues arrangement of the matrix $\boldsymbol{A}^1$ ('x' symbols) and eigenvalues arrangement of the matrix $\boldsymbol{\Psi}\boldsymbol{A}^1\boldsymbol{\Psi}^-$ (diamonds). To keep neural networks stable, all eigenvalues of $\boldsymbol{\Psi}\boldsymbol{A}^1\boldsymbol{\Psi}^-$ should be stable. As one can observe in Figs. 10(b), (d) and (f), in the case considered all neurons are stable. To carry this out, the proposed constrained procedure keeps eigenvaules of $\boldsymbol{A}^1$ stable (Figs. 10 (b), (d) and (f)) and checks relations between feedforward filter parameters $\boldsymbol{b}_i^1$ (Condition 3 of Theorem 2). Table 1 shows the feedforward filter parameters values, which are correct. This proves that neurons with IIR filters are stable.

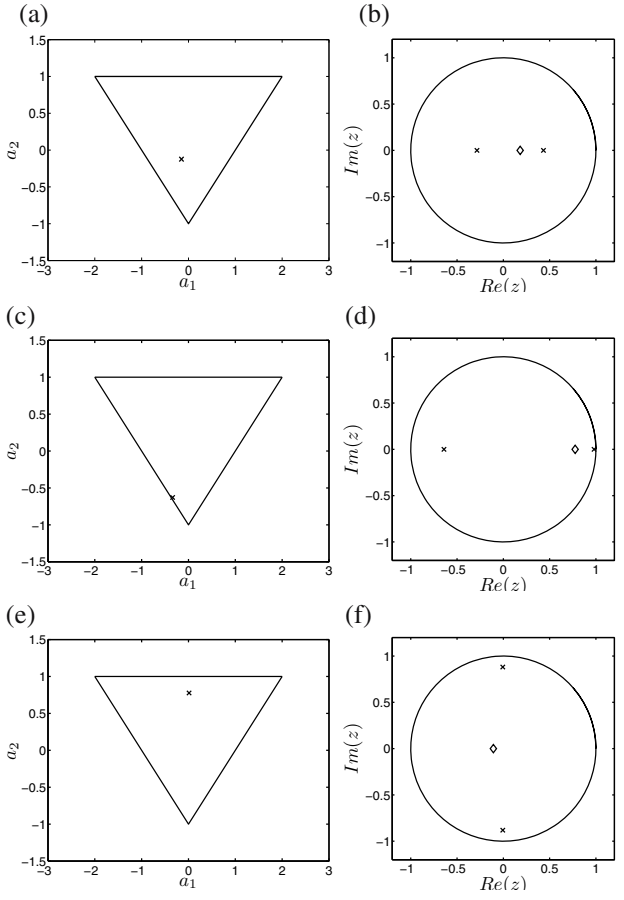

Fig. 10. Stabilization results—neurons with IIR filters. Parameters location: neuron 1 (a), neuron 2 (c), neuron 3 (e); Pole placement: neuron 1 (b), neuron 2 (d), neuron 3 (f).

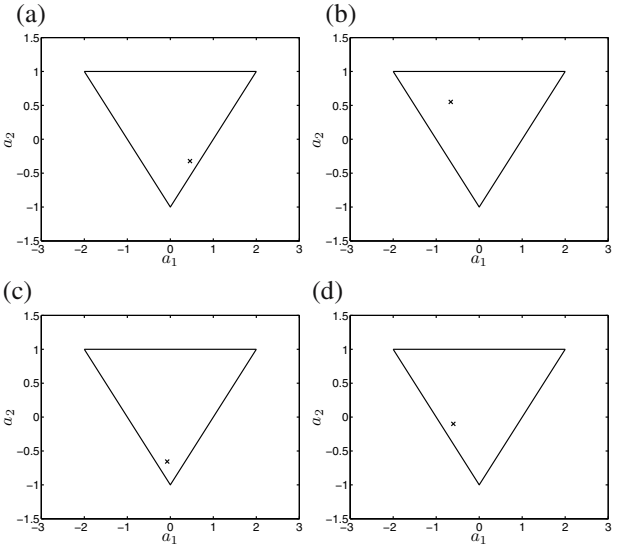

Fig. 11. Stabilization results—neurons with FIR filters. Parameters location: neuron 1 (a), neuron 2 (b), neuron 3 (c), neuron 4 (d).

In turn, in Fig. 11 one can see the location of the feedback filter parameters of the neurons with FIR filters (represented by the matrix $\boldsymbol{A}^2$) in the stability triangle. Parametres of each neuron are feasible. According to Theorem 2, the overall neural model is stable. The stabilization of the dynamic neural network based on constrained optimization and gradient projection works pretty well. After training, the neural model was tested checking modelling quality. The outputs of the neural model (dashed line) and the separately excited motor (solid line) generated for another 1000 testing samples are depicted in Fig. 12. The efficiency of the neural model was also checked during the work of the motor in closed-loop control. The results are presented in Fig. 13. For clarity of presentation of the modelling results, the outputs of the process and the neural model for about 250 time steps are only illustrated.

As one can see there, the output of the model tracks the changes of the process but the model output variance is significant. This could indicate that the neural model is too complex to model the system properly. Table 3 includes the quality measures in the form of the Sum of Squared Errors (SSE) and Mean Squared Errors (MSE) for both testing sets T_o (open-loop control) and T_c (closed-loop control). The experiment was repeated using the simple ARS algorithm. Unfortunately, in this case the neural model quickly lost stability and the error called the `floating point overflow` was generated. Summarizing, serious numerical problems were observed when the network was trained without stability considerations.

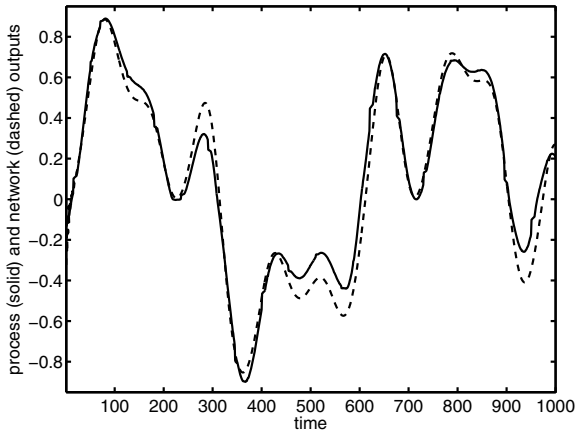

Fig. 12. Responses of the motor (solid) and the neural model (dashed)—open-loop control.

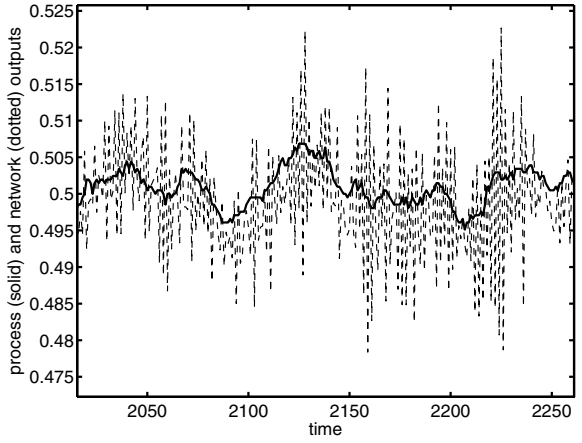

Fig. 13. Responses of the motor (solid) and the neural model (dashed)—closed-loop control.

4.4. Experiment 2. The neural network model (5) had the following structure: one input, five IIR neurons with first order filters and sigmoidal activation functions, four FIR neurons with first order filters and linear activation functions, and one linear output neuron. In all cases considered, the training process was carried out for 100 steps using the ARS algorithm with the initial variance $v_0 = 0.1$. Firstly, ARS with gradient projection was employed and then experiment was repeated using the simple ARS algorithm. According to Theorem 1, to guarantee the stability of the neural model it is enough to keep the absolute value of feedback filter parameters less than one. As one can see in Table 2, the training procedure keeps all

Table 2. Feedback filter parameters.

i	1	2	3	4	5
a_{1i}^1	0.4191	-0.2098	0.4328	-0.1536	-0.3965
a_{1i}^2	-0.3993	-0.9819	0.1342	-0.3116	–

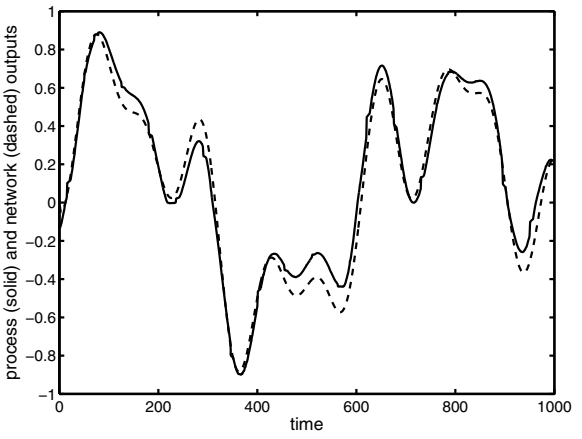

Fig. 14. Responses of the motor (solid) and the neural model (dashed)—open-loop control.

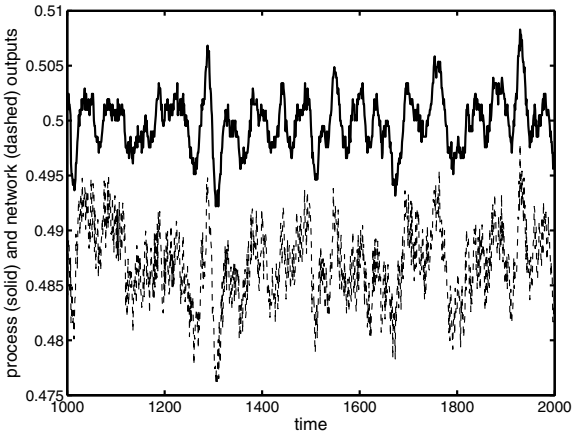

Fig. 15. Responses of the motor (solid) and the neural model (dashed)—close-loop control.

feedback filter parameters inside the interval $(-1, 1)$ and the model is stable. Once again, the stabilization of the dynamic neural network based on constrained optimization and gradient projection works pretty well.

After training, the neural model was tested checking modelling quality. The outputs of the neural model (dashed line) and the separately excited motor (solid line) generated for another 1000 testing samples are depicted in Fig. 14. This time the model is simpler but modelling accuracy is slightly better than in the previously discussed case (Table 3). The efficiency of the neural model was also checked during the work of the motor in the closed-loop control. The results are presented in Fig. 15. For the clarity of presentation of the modelling results, the outputs of the process and the neural model for 1000 time steps are only illustrated. It is observed that the variance of the model output is much lower than in the previous case, but the bias is bigger. This could indicate that the neural model is too simple to model the system properly.

Table 3. Modelling quality indices.

	Experiment 1		Experiment 2	
	set T_o	set T_c	set T_o	set T_c
SSE	6.9875	0.1893	6.2659	0.5649
MSE	0.007	6.310^{-5}	0.0063	1.910^{-4}

The bias–variance trade-off is strongly related to the problem of model selection, but this issue is out of the scope of this paper.

5. Conclusions

The purpose of this paper was to propose a method for stability analysis of the cascade locally recurrent neural model. To tackle this problem, Lyapunov's first method was applied. Using this method, local stability conditions were derived. The stability conditions rendered it possible to define feasible regions for the network parameters and then, using the gradient projection, a stable training algorithm. The method was checked using a number of experiments, showing its usefulness and efficiency. It should be pointed out that the methods are very simple and numerically uncomplicated, and they can be easily introduced into the learning procedure. The example of the identification of a real process confirms the effectiveness of the proposed learning with stabilization.

Acknowledgment

This work was supported in part by the Ministry of Science and Higher Education in Poland under Grant No. N514 1219 33.

References

Back, A. D. and Tsoi, A. C. (1991). FIR and IIR synapses, A new neural network architecture for time series modelling, *Neural Computation* **3**(3): 375–385.

Campolucci, P. and Piazza, F. (2000). Intrinsic stability-control method for recursive filters and neural networks, *IEEE Transactions on Circuit and Systems—II: Analog and Digital Signal Processing* **47**(8): 797–802.

Cannas, B., Cincotti, S., Marchesi, M. and Pilo, F. (2001). Learnig of Chua's circuit attractors by locally recurrent neural networks, *Chaos Solitons & Fractals* **12**(11): 2109–2115.

Cao, J., Yuan, K. and Li, H. (2006). Global asymptotical stability of recurrent neural networks with multiple discrete delays and distributed delays, *IEEE Transactions on Neural Networks* **17**(6): 1646–1651.

Ensari, T. and Arik, S. (2005). Global stability analysis of neural networks with multiple time varying delays, *IEEE Transactions on Automatic Control* **50**(11): 1781–1785.

Fasconi, P., Gori, M. and Soda, G. (1992). Local feedback multilayered networks, *Neural Computation* **4**(1): 120–130.

Forti, M., Nistri, P. and Papini, D. (2005). Global exponential stability and global convergence in finite time of delayed neural networks with infinite gain, *IEEE Transactions on Neural Networks* **16**(6): 1449–1463.

Gori, M., Bengio, Y. and Mori, R. D. (1989). BPS: A learning algorithm for capturing the dynamic nature of speech, *International Joint Conference on Neural Networks, Washington DC, USA*, Vol. II, pp. 417–423.

Gupta, M. M., Jin, L. and Homma, N. (2003). *Static and Dynamic Neural Networks. From Fundamentals to Advanced Theory*, John Wiley & Sons, Hoboken, NJ.

Gupta, M. M. and Rao, D. H. (1993). Dynamic neural units with application to the control of unknown nonlinear systems, *Journal of Intelligent and Fuzzy Systems* **1**(1): 73–92.

Marcu, T., Mirea, L. and Frank, P. M. (1999). Development of dynamical neural networks with application to observer based fault detection and isolation, *International Journal of Applied Mathematics and Computer Science* **9**(3): 547–570.

Patan, K. (2007). Stability analysis and the stabilization of a class of discrete-time dynamic neural network, *IEEE Transactions on Neural Networks* **18**(3): 660–673.

Patan, K. (2008a). Aproximation of state-space trajectories by locally recurrent globally feed-forward neural networks, *Neural Networks* **21**(1): 59–64.

Patan, K. (2008b). *Artificial Neural Networks for the Modelling and Fault Diagnosis of Technical Processes*, Lecture Notes in Control and Information Sciences, Vol. 377, Springer–Verlag, Berlin.

Patan, K. (2008c). Stability criteria for three-layer locally recurrent networks, *Proceedings of the 17th IFAC World Congress on Automatic Control, Seoul, Korea,* (on CD-ROM).

Patan, K. and Parisini, T. (2005). Identification of neural dynamic models for fault detection and isolation: The case of a real sugar evaporation process, *Journal of Process Control* **15**(1): 67–79.

Patan, K., Witczak, M. and Korbicz, J. (2008). Towards robustness in neural network based fault diagnosis, *International Journal of Applied Mathematics and Computer Science* **18**(4): 443–454, DOI: 10.2478/v10006-008-0039-2.

Tsoi, A. C. and Back, A. D. (1994). Locally recurrent globally feedforward networks: A critical review of architectures, *IEEE Transactions on Neural Networks* **5**(2): 229–239.

Xiang-Qun, L. and Zhang, H. Y. (2000). Fault detection and diagnosis of permanent-magnet DC motor based on parameter estimation and neural network, *IEEE Transactions on Industrial Electronics* **47**(5): 1021–1030.

Zhang, J., Morris, A. J. and Martin, E. B. (1998). Long term prediction models based on mixed order locally recurrent neural networks, *Computers Chemical Engineering* **22**(7–8): 1051–1063.

Krzysztof Patan was born in 1971 in Zielona Góra, Poland. He received the M.Sc. degree in electrical engineering from the Technical University of Zielona Góra, Poland, in 1996, the Ph.D. degree in machine design and exploitation from the Warsaw University of Technology, Poland, in 2000, and the D.Sc. degree in electrical engineering from the University of Zielona Góra, Poland, in 2009. Currently, he is an assistant professor at the Institute of Control and Computation Engineering, University of Zielona Góra. His research interests include artificial neural networks and their application to modelling and identification of nonlinear systems, fault detection and diagnosis, fault tolerant control systems, and optimization techniques.

Received: 4 February 2009
Revised: 14 October 2009