10.1515/acsc-2016-0007

Archives of Control Sciences
Volume 26(LXII), 2016
No. 1*, pages 117–142*

Elman neural network for modeling and predictive control of delayed dynamic systems

ANTONI WYSOCKI and MACIEJ ŁAWRYŃCZUK

The objective of this paper is to present a modified structure and a training algorithm of the recurrent Elman neural network which makes it possible to explicitly take into account the time-delay of the process and a Model Predictive Control (MPC) algorithm for such a network. In MPC the predicted output trajectory is repeatedly linearized on-line along the future input trajectory, which leads to a quadratic optimization problem, nonlinear optimization is not necessary. A strongly nonlinear benchmark process (a simulated neutralization reactor) is considered to show advantages of the modified Elman neural network and the discussed MPC algorithm. The modified neural model is more precise and has a lower number of parameters in comparison with the classical Elman structure. The discussed MPC algorithm with on-line linearization gives similar trajectories as MPC with nonlinear optimization repeated at each sampling instant.

Key words: dynamic models, process control, model predictive control, neural networks, Elman neural network, delayed systems.

1. Introduction

Neural networks are often used for modeling dynamic processes [9, 11, 22, 25]. In most cases the Multi-Layer Perceptron (MLP) network with one hidden layer is used, the network with Radial Basis Functions (RBF) is used much less frequently. Both MLP and RBF neural structures are in fact static approximators. When they are used for modeling of dynamic processes (in the discrete-time domain), the dynamics is introduced into the static networks by delivering the values of the input and output signals from some previous sampling instants to the input nodes of the network (by the delay lines). Such an approach, although very frequently used in practice, e.g. [14, 16], may need relatively high dynamic order, i.e. the number of input and output signals from previous instants may be significant [23].

An interesting alternative to static neural approximators with delay lines is to use recurrent neural networks [11, 21, 25, 26], which are dynamic models by nature. One well-known example of recurrent neural networks is Elman neural network [5, 21, 25].

The Authors are with Institute of Control and Computation Engineering, Faculty of Electronics and Information Technology, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland. E-mails: A.T.Wysocki@stud.elka.pw.edu.pl, M.Lawrynczuk@ia.pw.edu.pl

Received 06.10.2015. Revised 03.03.2016.

The Elman networks are used in various fields, e.g. they may be used to estimate pilot workload [8], to monitor condition in nuclear power plant and rotating machinery [29], to predict geomagnetic storms from solar wind data [31], for electric load forecasting [32], for short-term temperature forecasting [17], for modeling the flow of passengers in subway [18] and for identification of the grammatical structure of literary works [19]. A comparison of MLP, RBF and Elman dynamic models is given in [4].

Because the Elman neural network is capable of precise modeling of dynamic processes, it may be used for control purposes. In particular, the Elman network is used in Model Predictive Control (MPC): e.g. MPC of two benchmark dynamic systems is considered in [4], air pressure control supplied to the disc drill subway tunnel under a river is reported in in [34], control of watertanks is described in [28] and control of autonomous underwater vehicle is demonstrated in [33]. In all the cited works the same approach to MPC is used, in which a nonlinear optimization routine finds on-line the optimal control sequence (the Levenberg-Marquardt algorithm is used in the first case, a heuristic particle swarm optimization technique is used in the second case, a quasi-Newton algorithm is used in the third system and an inefficient steepest-descent method is used in the last application). Although on-line nonlinear optimization works in simulations, in real control applications it may cause problems resulted from its inherent computational difficulty. Furthermore, it is practically impossible to guarantee or even check that the nonlinear optimization algorithm finds a global solution.

In the classical recurrent Elman neural network [5] there is no extra time-delay, while many technological dynamic processes, in particular in chemical, petrochemical and food industries, are characterized by relatively long pure time-delay, which may result from the time necessary to perform the measurements. Intuitively, in such cases the use of the classical Elman neural network (i.e. with no delay) may be not the best solution. One may expect that the classical Elman network needs quite many hidden nodes (and weights) and its prediction accuracy may be below expectations.

This work:

a) presents a modified structure of the recurrent Elman neural network which makes it possible to explicitly take into account the pure time-delay of the process and describes a training algorithm of the modified network,

b) details derivation and implementation of an MPC algorithm in which such a modified Elman network is used.

In contrast to all cited works concerned with MPC based on the classical Elman network, the discussed MPC algorithm does not need computationally demanding and possibly troublesome on-line nonlinear optimization. Conversely, the predicted output trajectory is repeatedly linearized on-line along the future input trajectory, which makes to possible to calculate the optimal control sequence from an easy to solve quadratic optimization problem. A nonlinear simulated neutralization reactor is considered to show advantages of the modified Elman neural network and the discussed MPC algorithm. The modified

neural model is compared with the classical Elman structure in terms of complexity and accuracy. Furthermore, the discussed MPC algorithm with on-line linearization is compared with the general nonlinear MPC approach with nonlinear optimization repeated at each sampling instant.

2. Elman neural networks

For simplicity and clarity of presentation single-input single-output dynamic systems are considered in this work. The input signal is denoted by u, the output signal is denoted by y. The current sampling instant is denoted by k.

2.1. Classical Elman recurrent neural network

Fig. 1 shows the structure of the classical Elman neural network. The network has one input associated with the input signal of the process from the previous sampling instant, i.e. $u(k-1)$, K hidden neurons with nonlinear transfer function $\varphi : R \to R$, one neuron (adder) and one output $y(k)$. Output signals of the hidden layer $(v_1(k),\ldots,v_K(k))$ are entered through single delay blocks to the input nodes of the network, which means that the network has $K+1$ input nodes $u(k-1), v_1(k-1),\ldots,v_K(k-1)$. The weights of the second layer of the network are denoted by $w_i^{(2)}$ for $i = 0,\ldots,K$, the weights of the first layer of the network $w_{i,j}^{(1)}$ for $i = 1,\ldots,K$, $j = 0,\ldots,K+1$. Both layers have a bias signal, i.e. an additional unity signal.

2.2. Modified Elman recurrent neural network

Numeous dynamic systems, in particular in chemical, petrochemical and food industries, are characterized by relatively long pure time-delay, which may result from the time necessary to perform the measurements. A straightforward way to take into account that fact is to incorporate the delay into the recurrent Elman neural network. Fig. 2 shows the structure of the modified Elman neural network. In the modified network the input signal of the model $(u(k-\tau))$ explicitly takes into account the pure time-delay τ whereas in the classical structure the neural model itself must approximate the delayed nature of the dynamic system. The weights of the network are denoted in the same way as in the case of the classical Elman structure. The output of the modified Elman neural network model is

$$y_{\mathrm{mod}}(k) = w_0^{(2)} + \sum_{i=0}^{K} w_i^{(2)} v_i(k) \tag{1}$$

where $v_i(k)$ stands for the output signals of consecutive hidden nodes $(i = 1,\ldots,K)$, $v_0(k) = 1$. One has

$$v_i(k) = \varphi(z_i(k)) \tag{2}$$

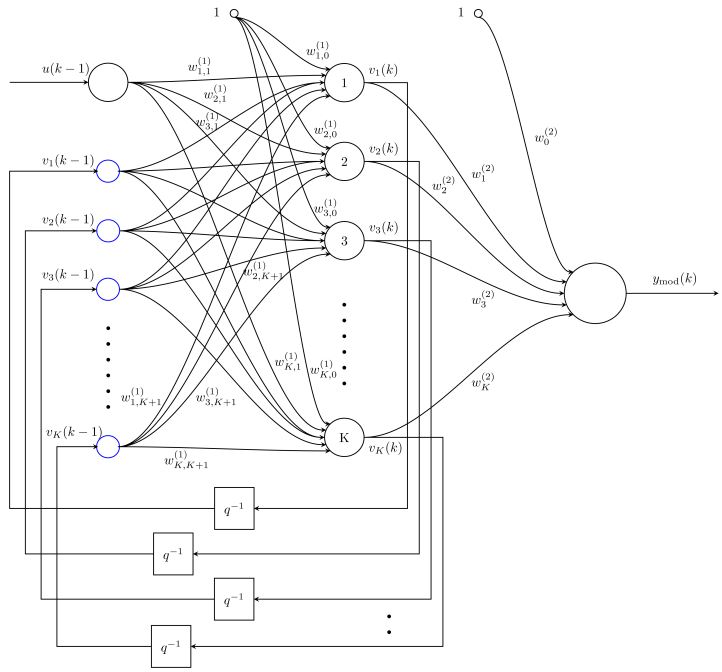

Figure 1. Structure of classical Elman neural network

where the sum of signals of the i^{th} hidden node is

$$z_i(k) = w_{i,0}^{(1)} + w_{i,1}^{(1)} u(k-\tau) + \sum_{j=1}^{K} w_{i,j+1}^{(1)} v_j(k-1) \tag{3}$$

Combining Eqs. (1), (2) and (3), one has

$$y_{\mathrm{mod}}(k) = w_0^{(2)} + \sum_{i=0}^{K} w_i^{(2)} \varphi\left(w_{i,0}^{(1)} + w_{i,1}^{(1)} u(k-\tau) + \sum_{j=1}^{K} w_{i,j+1}^{(1)} v_j(k-1) \right) \tag{4}$$

2.3. Gradient-based training of modified recurrent Elman neural networks

Training of the neural network consists in modifying the values of its parameters (weights) in such a way that the approximation error is acceptable. In the considered case of identification of dynamic processes, sequences of recorded data of input and output signals are necessary for training. Model error (approximation accuracy) is usually defined by the sum of squared errors. For some vector of weights $\boldsymbol{w}$, one has

$$E(\boldsymbol{w}) = \sum_{k=1}^{P} (y_{\mathrm{mod}}(k) - y(k))^2 \tag{5}$$

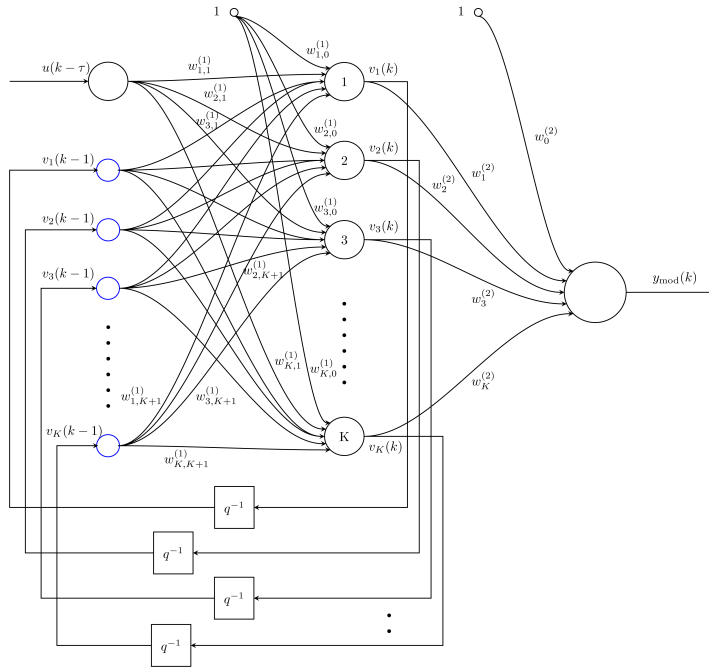

Figure 2. Structure of modified Elman neural network

where $y_{\mathrm{mod}}(k)$ is the output of the neural network, $y(k)$ is an output signal of the real process (the training pattern recorded in the data set), P is the number of data patterns. The training process consists in minimising the model error function (5). Due to the nonlinear hidden layer transfer function φ, this is an unconstrained nonlinear optimization problem. Because nonlinear optimization may be difficult and give local solutions, training is usually repeated many times starting with different initial weights (the multi-start approach).

The general gradient-based training algorithm, leading to minimization of the model error function (5) may be summarized in the following steps (the consecutive iterations are denoted by $it = 1, \ldots, it^{\max}$):

0. Initialization of the weights $\boldsymbol{w}$, random values are usually assumed from the range $< -1, 1 >$.

1. Calculation of the model output signal $y_{\mathrm{mod}}(k)$ for all sampling instants ($k = 1, \ldots, P$) for the current weights. For the modified Elman network Eqs. (1), (2), (3) and (4) are used.

2. Calculation of the model error from Eq. (5).

3. If the model error or a norm of its gradient satisfies a stopping criterion, the algorithm is stopped.

4. Calculation of the optimization direction $\boldsymbol{p}_{it}$.
5. Calculation of the optimal step-length η_{it} along the direction $\boldsymbol{p}_{it}$.
6. The model weights are updated $\boldsymbol{w}_{it+1} = \boldsymbol{w}_{it} + \eta_{it}\boldsymbol{p}_{it}$, the training algorithm goes to step 1.

The simplest approach to find the optimization direction is to use the steepest-descent technique, in which the direction is opposite to the current gradient of the model error [24], i.e.

$$\boldsymbol{p}_{it} = -\frac{\mathrm{d}E(\boldsymbol{w}_{it})}{\mathrm{d}\boldsymbol{w}_{it}}$$

Due to very slow convergence of the steepest-descent method, a quasi-Newton algorithms [24] are recommended in this work in which the direction is calculated from the general formula

$$\boldsymbol{p}_{it} = -[\boldsymbol{H}(\boldsymbol{w}_{it})]^{-1}\frac{\mathrm{d}E(\boldsymbol{w}_{it})}{\mathrm{d}\boldsymbol{w}_{it}}$$

where $\boldsymbol{H}(\boldsymbol{w}_{it})$ is Hessian matrix of the error function $E(\boldsymbol{w})$, i.e. $\boldsymbol{H}(\boldsymbol{w}_{it}) = \frac{\mathrm{d}^2E(\boldsymbol{w})}{\mathrm{d}\boldsymbol{w}^2}$. Because analytical calculation of the inverse of the Hessian matrix is quite complex, it is not found analytically, but approximated. In this work a very efficient Broyden-Fletcher-Goldfarb-Shanno (BFGS) method is used. In each iteration of the algorithm the inverse Hessian $[\boldsymbol{H}(\boldsymbol{w}_{it})]^{-1}$ is approximated by the matrix $\boldsymbol{V}_{it}$ from the formula

$$\boldsymbol{V}_{it} = \boldsymbol{V}_{it-1} + \frac{\boldsymbol{s}_{it}\boldsymbol{s}_{it}^T}{\boldsymbol{s}_{it}^T\boldsymbol{r}_{it}} - \frac{\boldsymbol{V}_{it-1}\boldsymbol{r}_{it}\boldsymbol{r}_{it}^T\boldsymbol{V}_{it-1}}{\boldsymbol{r}_{it}^T\boldsymbol{V}_{it-1}\boldsymbol{r}_{it}},$$

where the increment of the gradient vector of the weights vector is denoted by $\boldsymbol{r}_{it} = \frac{\mathrm{d}E(\boldsymbol{w}_{it})}{\mathrm{d}\boldsymbol{w}_{it}} - \frac{\mathrm{d}E(\boldsymbol{w}_{it-1})}{\mathrm{d}\boldsymbol{w}_{it-1}}$, whereas the increment of the weights vector is $\boldsymbol{s}_{it} = \boldsymbol{w}_{it} - \boldsymbol{w}_{it-1}$. The gradients of the error function are determined analytically at each iteration of the training algorithm. Differentiating Eq. (5) with respect to the weights of the first and the second layers, one obtains

$$\frac{\mathrm{d}E(\boldsymbol{w})}{\mathrm{d}w_{i,j}^{(1)}} = 2(y_{\mathrm{mod}}(k) - y(k))\frac{\partial y_{\mathrm{mod}}(k)}{\partial w_{i,j}^{(1)}}$$

for all $i = 1\ldots K,\ j = 0,\ldots,K+1$ and

$$\frac{\mathrm{d}E(\boldsymbol{w})}{\mathrm{d}w_{i}^{(2)}} = 2(y_{\mathrm{mod}}(k) - y(k))\frac{\partial y_{\mathrm{mod}}(k)}{\partial w_{i}^{(2)}}$$

for all $i = 0,\ldots,K$. Next, differentiating Eq. (4), one has

$$\frac{\partial y_{\mathrm{mod}}(k)}{w_{i,j}^{(1)}} = \sum_{n=1}^{K} w_n^{(2)}\frac{\partial v_n(k)}{\partial w_{i,j}^{(1)}}$$

for all $i = 0 \ldots K,\ j = 0 \ldots K+1$. For the second layer

$$\frac{\partial y_{\mathrm{mod}}(k)}{\partial w_i^{(2)}} = v_i(k)$$

for all $i = 0, \ldots, K$. Differentiating Eq. (2) gives

$$\frac{\partial v_n(k)}{\partial w_{i,j}^{(1)}} = \frac{\mathrm{d}\varphi(z_n(k))}{\mathrm{d}z_n(k)} \frac{\partial z_n(k)}{\partial w_{i,j}^{(1)}}$$

for all $i = 1, \ldots, K,\ j = 0, \ldots, K+1,\ n = 1, \ldots, K$. Finally, from Eq. (3), one has

$$\frac{\partial z_n(k)}{\partial w_{i,j}^{(1)}} = \begin{cases} \displaystyle\sum_{p=1}^{K} w_{n,n_u+p}^{(1)} \frac{\partial v_p(k-1)}{\partial w_{i,j}^{(1)}} & \text{for } i = n \\ X_j(k-1) \displaystyle\sum_{p=1}^{K} w_{n,n_u+p}^{(1)} \frac{\partial v_p(k-1)}{\partial w_{i,j}^{(1)}} & \text{for } i \neq n \end{cases} \tag{6}$$

for all $i = 0, \ldots, K,\ j = 0, \ldots, K+1,\ n = 1, \ldots, K$, where

$$X_j(k-1) = \begin{cases} 1 & \text{for } j = 0 \\ u(k-\tau) & \text{for } j = 1 \\ v_j(k-1) & \text{for } 1 < j \leqslant K+1 \end{cases}$$

Calculation of the optimal step-length η_{it} along the minimization direction $\boldsymbol{p}_{it}$ may be done by means of many methods, e.g. the golden section approach or the Armijo's rule [24].

3. Model Predictive Control based on the modified Elman network

3.1. Mathematical formulation of MPC

The core idea of Model Predictive Control (MPC) algorithms is to use on-line a dynamic model of the controlled process to calculate some predicted control errors and to minimise a predefined cost-function which defines the future control quality [2, 20, 30]. In comparison with the classical single-loop PID controller, the MPC algorithms have the following advantages:

a) the ability to take into account constraints imposed on input and output variables (or state variables) in a systematic way,

b) the ability to control multi-input multi-output processes,

c) very good control quality.

MPC algorithms have many applications, mainly in process control (e.g. in petrochemical, chemical, food and paper industries) [27], but they are also used in case of less typical processes, e.g. in medicine for glucose concentration control [12] or in biology research for control of physiology in a free living cell [1].

From the mathematical point of view, at each consecutive sampling instant $k = 1,2,\ldots$ of the MPC algorithm the following set of future control increments

$$\triangle \boldsymbol{u}(k) = [\triangle u(k|k) \ \ldots \ \triangle u(k+N_{\mathrm{u}}-1|k)]^{\mathrm{T}} \in \mathbb{R}^{N_{\mathrm{u}}} \tag{7}$$

is calculated, where $\triangle u(k+p|k) = u(k+p|k) - u(k+p-1|k)$, N_{u} is the control horizon, in such a way that the predicted control errors are minimized over the prediction horizon N. Typically, the quadratic cost-function is used for optimization

$$J(k) = \sum_{p=1}^{N} (y^{\mathrm{sp}}(k+p|k) - \hat{y}(k+p|k))^2 + \lambda \sum_{p=0}^{N_{\mathrm{u}}-1} (\triangle u(k+p|k))^2 \tag{8}$$

where $y^{\mathrm{sp}}(k+p|k)$ denotes the set-point for the sampling instant $k+p$ known in the current instant k, $\hat{y}(k+p|k))$ is the prediction of the output signal for the sampling instant $k+p$ predicted in the current instant k, λ is a weighting factor. Predicted signals $\hat{y}(k+p|k))$ are calculated over the prediction horizon, i.e. for $p = 1,\ldots,N$, from a dynamic model of the controlled process. Although at each sampling instant the whole vector (7) of the increments of the manipulated variable is calculated, only its first element is applied to the process and in the next sampling instant the whole procedure is repeated.

In general, it is possible to take into account some constraints imposed on manipulated and controlled variables. Let $u^{\min}$, $u^{\max}$, $\triangle u^{\max}$ define the constraints imposed on the minimal value, the maximal value and the rate of change of the manipulated variable and let $y^{\min}$, $y^{\max}$ define the constraints imposed on the minimal value and maximal values of the predicted output variable. Using the cost-function (8), the MPC optimization problem solved at each sampling instant is

$$\min_{\triangle u(k|k),\ldots,\triangle u(k+N_{\mathrm{u}}-1|k)} \left\{ J(k) = \sum_{p=1}^{N} \|y^{\mathrm{sp}}(k+p|k) - \hat{y}(k+p|k)\|^2 + \lambda \sum_{p=0}^{N_{\mathrm{u}}-1} \|\triangle u(k+p|k)\|^2 \right\}$$

subject to

$$\begin{aligned}
&u^{\min} \leqslant u(k+p|k) \leqslant u^{\max}, \text{ for } p = 0,\ldots,N_{\mathrm{u}}-1 \\
&-\triangle u^{\max} \leqslant \triangle u(k+p|k) \leqslant \triangle u^{\max}, \text{ for } p = 0,\ldots,N_{\mathrm{u}}-1 \\
&y^{\min} \leqslant y(k+p|k) \leqslant y^{\max}, \text{ for } p = 1,\ldots,N
\end{aligned} \tag{9}$$

In the matrix-vector form the optimization problem becomes

$$\min_{\triangle \boldsymbol{u}(k)} \left\{ J(k) = \|\boldsymbol{y}^{\mathrm{sp}}(k) - \hat{\boldsymbol{y}}(k)\|^2 + \|\triangle \boldsymbol{u}(k)\|^2_{\boldsymbol{\Lambda}} \right\}$$

subject to

$$\begin{aligned}
&\boldsymbol{u}^{\min} \leqslant \boldsymbol{u}(k) \leqslant \boldsymbol{u}^{\max} \\
&-\triangle\boldsymbol{u}^{\max} \leqslant \triangle\boldsymbol{u}(k) \leqslant \triangle\boldsymbol{u}^{\max} \\
&\boldsymbol{y}^{\min} \leqslant \hat{\boldsymbol{y}}(k) \leqslant \boldsymbol{y}^{\max}
\end{aligned} \tag{10}$$

where the vectors of the set-point and the predicted trajectories are

$$\begin{aligned}
\boldsymbol{y}^{\mathrm{sp}}(k) &= [y^{\mathrm{sp}}(k) \ \ldots \ y^{\mathrm{sp}}(k)]^{\mathrm{T}} \in \mathbb{R}^N \\
\hat{\boldsymbol{y}}(k) &= [\hat{y}(k+1|k)(k) \ \ldots \ \hat{y}(k+N|k)(k)]^{\mathrm{T}} \in \mathbb{R}^N
\end{aligned}$$

the input constraints are defined by the vectors

$$\begin{aligned}
\boldsymbol{u}^{\min} &= \left[u^{\min} \ \ldots \ u^{\min}\right]^{\mathrm{T}} \in \mathbb{R}^{N_{\mathrm{u}}} \\
\boldsymbol{u}^{\max} &= [u^{\max} \ \ldots \ u^{\max}]^{\mathrm{T}} \in \mathbb{R}^{N_{\mathrm{u}}} \\
\triangle\boldsymbol{u}^{\max} &= [\triangle u^{\max} \ \ldots \ \triangle u^{\max}]^{\mathrm{T}} \in \mathbb{R}^{N_{\mathrm{u}}}
\end{aligned}$$

the output constraints are defined by the vectors

$$\begin{aligned}
\boldsymbol{y}^{\min} &= \left[y^{\min} \ \ldots \ y^{\min}\right]^{\mathrm{T}} \in \mathbb{R}^N \\
\boldsymbol{y}^{\max} &= [y^{\max} \ \ldots \ y^{\max}]^{\mathrm{T}} \in \mathbb{R}^N
\end{aligned}$$

and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda, \ldots, \lambda)$ is a matrix of dimensionality $N_{\mathrm{u}} \times N_{\mathrm{u}}$.

The most intuitive approach to nonlinear MPC is to use a nonlinear model of the process, e.g. the Elman neural network, for calculation of the predicted output signals. Since the model is nonlinear, future predictions are nonlinear functions of the calculated control increments (7). As a result, the MPC optimization problem (9) or (10) is in fact a nonlinear, in general non-convex, task which must be solved in real time on-line at each sampling instant. Computational complexity of such an approach may be high and the optimization algorithm may be unable to find the solution within the required time.

3.2. MPC algorithm with nonlinear prediction and linearization along the predicted trajectory (MPC-NPLPT)

The general idea of a simple method which leads to reduction of computational burden of nonlinear MPC is quite straightforward: at each sampling instant the nonlinear model of the process is linearized on-line for the current operating conditions of the process and the obtained linear approximation (i.e. a linear model with time-varying parameters) is used for prediction. Linearization makes it possible to obtain a relatively simple to solve quadratic optimization problem. Such a simple approach to nonlinear MPC is effective for the processes with mild nonlinearities whereas for highly nonlinear systems the simplest solution may be inadequate as demonstrated in [16]. It is because the same linearized model is used for prediction over the whole prediction horizon. When the set-point changes significantly, the predictions differ considerably from the real process output values.

An alternative approach to nonlinear MPC is to use for prediction not a linear approximation of the model obtained for the current operating conditions, but to directly determine a linear approximation of the predicted output trajectory along the future (calculated) input trajectory [16]. Similarly to MPC with simple model linearization, the MPC algorithm with Nonlinear Prediction and Linearization along the Predicted Trajectory (MPC-NPLPT) also leads to a quadratic optimization problem, on-line nonlinear optimization is not necessary. In order to increase prediction (and control) accuracy, trajectory linearization and optimization of the future control sequence are repeated a few times in internal iterations of the algorithm. Let the internal iterations be denoted by the superscript t (for $t = 1,2,3,\ldots,t_{\max}$). Using the Taylor's series expansion, the linear approximation of the nonlinear output trajectory $\hat{\boldsymbol{y}}^t(\boldsymbol{u}^t(k))$ is

$$\hat{\boldsymbol{y}}^t(k) = \hat{\boldsymbol{y}}^{t-1}(k) + \boldsymbol{H}^t(k)(\boldsymbol{u}^t(k) - \boldsymbol{u}^{t-1}(k)) \tag{11}$$

In Eq. (11) the vector $\boldsymbol{u}^t(k) = [u^t(k|k)\ldots u^t(k+N_{\mathrm{u}}-1|k)]^{\mathrm{T}}$ consists of the future control sequence which is calculated in the current sampling instant k and the current internal iteration t (i.e. the decision variables of MPC). Linearization is performed along the known future control sequence most recently calculated, from the previous internal iteration, i.e. $\boldsymbol{u}^{t-1}(k) = \left[u^{t-1}(k|k)\ldots u^{t-1}(k+N_{\mathrm{u}}-1|k)\right]^{\mathrm{T}}$. The predicted future output trajectory $\hat{\boldsymbol{y}}^{t-1}(k) = \left[\hat{y}^{t-1}(k+1|k)\ldots\hat{y}^{t-1}(k+N|k)\right]^{\mathrm{T}}$ is calculated from the dynamic model of the process for the input trajectory $\boldsymbol{u}^{t-1}(k)$ taking into account that $u^{t-1}(k+p|k) = u^{t-1}(k+N_{\mathrm{u}}-1|k)$ for $p = N_{\mathrm{u}},\ldots,N$. The matrix

$$\boldsymbol{H}^t(k) = \left.\frac{\mathrm{d}\hat{\boldsymbol{y}}(k)}{\mathrm{d}\boldsymbol{u}(k)}\right|_{\substack{\hat{\boldsymbol{y}}(k)=\hat{\boldsymbol{y}}^{t-1}(k)\\ \boldsymbol{u}(k)=\boldsymbol{u}^{t-1}(k)}} = \frac{\mathrm{d}\hat{\boldsymbol{y}}^{t-1}(k)}{\mathrm{d}\boldsymbol{u}^{t-1}(k)} = \begin{bmatrix} \dfrac{\partial\hat{y}^{t-1}(k+1|k)}{\partial u^{t-1}(k|k)} & \cdots & \dfrac{\partial\hat{y}^{t-1}(k+1|k)}{\partial u^{t-1}(k+N_{\mathrm{u}}-1|k)} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial\hat{y}^{t-1}(k+N|k)}{\partial u^{t-1}(k|k)} & \cdots & \dfrac{\partial\hat{y}^{t-1}(k+N|k)}{\partial u^{t-1}(k+N_{\mathrm{u}}-1|k)} \end{bmatrix} \tag{12}$$

is of dimensionality $N \times N_{\mathrm{u}}$ and it is calculated at each internal iteration of each sampling instant independently. The vector $\boldsymbol{u}^t(k)$ is determined for each internal iteration from

$$\boldsymbol{u}^t(k) = \boldsymbol{J}\triangle\boldsymbol{u}^t(k) + \boldsymbol{u}(k-1) \tag{13}$$

where $\boldsymbol{u}(k-1) = [u(k-1)\ldots u(k-1)]^{\mathrm{T}}$ and $\boldsymbol{J}$ is the all ones lower triangular matrix of dimensionality $N_{\mathrm{u}} \times N_{\mathrm{u}}$. In the first internal iteration ($t = 1$) of the MPC-NPLPT algorithm the input trajectory along which the output trajectory is linearized is taken from the previous sampling instant, i.e. $\boldsymbol{u}^0(k) = [u(k-1)\ldots u(k-1)]^{\mathrm{T}}$. Using Eq. (13), the linear approximation of the nonlinear predicted output trajectory (11) may be expressed as a function of the control increments calculated at the current sampling instant k and the current internal iteration t

$$\hat{\boldsymbol{y}}^t(k) = \boldsymbol{H}^t(k)\boldsymbol{J}\triangle\boldsymbol{u}^t(k) + \hat{\boldsymbol{y}}^{t-1}(k) + \boldsymbol{H}^t(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{t-1}(k)) \tag{14}$$

Using the obtained linearized prediction equation (14), the optimization problem (10) becomes the quadratic programming task in vector-matrix notation

$$\min_{\triangle \boldsymbol{u}^t(k)} \{J(k) = \|\boldsymbol{y}^{\mathrm{sp}}(k) - \boldsymbol{H}^t(k)\boldsymbol{J}\triangle\boldsymbol{u}^t(k) - \hat{\boldsymbol{y}}^{t-1}(k) - \boldsymbol{H}^t(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{t-1}(k))\|^2 + \|\triangle\boldsymbol{u}^t(k)\|^2_{\boldsymbol{\Lambda}}\}$$

subject to

$$\begin{aligned}
&\boldsymbol{u}^{\min} \leqslant \boldsymbol{J}\triangle\boldsymbol{u}^t(k) + \boldsymbol{u}(k-1) \leqslant \boldsymbol{u}^{\max} \\
&-\triangle\boldsymbol{u}^{\max} \leqslant \triangle\boldsymbol{u}^t(k) \leqslant \triangle\boldsymbol{u}^{\max} \\
&\boldsymbol{y}^{\min} \leqslant \boldsymbol{H}^t(k)\boldsymbol{J}\triangle\boldsymbol{u}^t(k) + \hat{\boldsymbol{y}}^{t-1}(k) + \boldsymbol{H}^t(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{t-1}(k)) \leqslant \boldsymbol{y}^{\max}
\end{aligned} \tag{15}$$

Internal iterations are continued if

$$\sum_{p=0}^{N_0} (y^{\mathrm{sp}}(k-p) - \hat{y}(k-p))^2 \geqslant \delta_y \tag{16}$$

where N_0 is a horizon, and a coefficient $\delta_y > 0$ is chosen experimentally. If the difference between the future control increment sequences calculated in two consecutive internal iterations is small, i.e. when

$$\|\triangle\boldsymbol{u}^t(k) - \triangle\boldsymbol{u}^{t-1}(k)\| < \delta_u \tag{17}$$

the internal iterations are interrupted, wherein the value of a coefficient δ_u is also chosen experimentally.

The steps repeated at each sampling instant k of the MPC-NPLPT algorithm are:

1. The first internal iteration ($t = 1$): the predicted output trajectory $\hat{\boldsymbol{y}}^0(k)$ corresponding to an assumed initial future input trajectory $\boldsymbol{u}^0(k)$ is determined from a nonlinear model of the process.

2. The nonlinear model is also used to calculate the linear approximation of the predicted trajectory $\hat{\boldsymbol{y}}^1(k)$ along the trajectory $\boldsymbol{u}^0(k)$ from Eq. (11) and the matrix $\boldsymbol{H}^1(k)$ from equation (12).

3. The quadratic programming task (15) is solved to find $\triangle\boldsymbol{u}^1(k)$.

4. If the condition (16) is satisfied, the internal iterations are continued for $t = 2, \ldots, t_{\max}$.

   4.1. The predicted output trajectory $\hat{\boldsymbol{y}}^{t-1}(k)$ corresponding to the input trajectory $\boldsymbol{u}^{t-1}(k) = \boldsymbol{J}\triangle\boldsymbol{u}^{t-1}(k) + \boldsymbol{u}(k-1)$ is calculated using the nonlinear model.

   4.2. The nonlinear model is also used to determine the linear approximation of the predicted output trajectory $\hat{\boldsymbol{y}}^t(k)$ along the trajectory $\boldsymbol{u}^{t-1}(k)$, i.e. the matrix $\boldsymbol{H}^t(k)$ is calculated.

4.3. The quadratic programming task (15) is solved to find control increments $\triangle \boldsymbol{u}^t(k)$ for the current internal iteration t and the sampling instant k.

4.4. If the condition (17) is satisfied or $t \geqslant t_{\max}$, the internal iterations are stopped. Otherwise, the internal iteration index is increased ($t := t+1$) and the algorithm goes to step 4.1.

5. The first element of the calculated vector of future control increments $\triangle \boldsymbol{u}^t(k)$ is applied to the process, i.e. $u(k) = u(k-1) + \triangle u^t(k|k)$.

6. In the next sampling instant ($k = k+1$) the algorithm goes to step 1.

3.3. Implementation of MPC-NPLPT algorithm for the modified Elman neural network

For the modified Elman neural network, using Eq. (4), the output predicted trajectory $\hat{\boldsymbol{y}}^t(k)$ corresponding to the input sequence $\boldsymbol{u}^t(k)$ is calculated in each internal iteration t of the MPC-NPLPT algorithm as

$$\hat{y}^t(k+p|k) = w_0^{(2)} + \sum_{i=1}^{K} w_i^{(2)} \varphi(z_i^t(k+p|k)) + d(k) \tag{18}$$

and from Eq. (3) the sum of signals of the i^{th} hidden node is

$$z_i^t(k+p|k) = w_{i,0}^{(1)} + w_{i,1}^{(1)} u^t(k-\tau+p|k) + \sum_{j=1}^{K} w_{i,j+1}^{(1)} v_j^t(k-1+p|k) \tag{19}$$

where from Eq. (2) $v_i^t(k+p|k) = \varphi(z_i^t(k+p|k))$. For prediction calculation in Eq. (18) the unmeasured disturbance acting on the process is assessed as the difference between the measured output signal of the process, $y(k)$, and the output of the model [30]

$$d(k) = y(k) - w_0^{(2)} \sum_{i=1}^{K} w_i^{(2)} \varphi\left(w_{i,0}^{(1)} + w_{i,1}^{(1)} u(k-\tau) + \sum_{j=1}^{K} w_{i,j+1}^{(1)} v_j(k-1) \right) \tag{20}$$

It is assumed that the disturbance $d(k)$ is constant over the whole prediction horizon. A linear approximation of the nonlinear output trajectory (Eq. (14)) is determined by a matrix $\boldsymbol{H}^t(k)$, each element of which is calculated by differentiating the predicted output trajectory (Eq. (18)) with respect to the future control sequence

$$\frac{\partial \hat{y}^t(k+p|k)}{\partial u^t(k-\tau+1+r|k)} = \sum_{i=1}^{K} w_i^{(2)} \frac{\partial v_i^t(k+p|k)}{\partial u^t(k-\tau+1+r+1|k)} \tag{21}$$

for $p = 1, \ldots, N$ and $r = 0, \ldots, N_{\text{u}} - 1$. The partial derivatives in the right side of Eq. (21) are calculated from

$$\frac{\partial v_i^t(k+p|k)}{\partial u^t(k-\tau+1+r+1|k)} = \frac{\mathrm{d}v_i^t(k+p|k)}{\mathrm{d}z_i^t(k+p|k)}\left(\frac{\partial u^t(k+p|k)}{\partial u^t(k-\tau+1+r+1|k)} + \frac{\partial z_i^t(k+p|k)}{\partial u^t(k-\tau+1+r+1|k)}\right) \tag{22}$$

Calculation of the first derivatives used in the right side of Eq. (22) depends on the type of the transfer function φ used in the hidden layer of the neural network. If the hyperbolic tangent ($\varphi(\cdot) = \tanh(\cdot)$) is used, one has

$$\frac{\mathrm{d}v_i^t(k+p|k)}{\mathrm{d}z_i^t(k+p|k)} = 1 - \tanh^2(z_i^t(k+p|k))$$

The first partial derivatives are calculated from

$$\frac{\partial u^t(k+p|k)}{\partial u^t(k-\tau+1+r+1|k)} = \begin{cases} w_{i,n+1}^1 & \text{if } p = r+1 \text{ or } (p > r+1 \text{ and } r = N_\mathrm{u} - 1) \\ 0 & \text{in other cases} \end{cases}$$

The second partial derivatives are determined from

$$\frac{\partial z_i^t(k+p|k)}{\partial u^t(k-\tau+1+r+1|k)} = \sum_{j=1}^{K} w_{i,j+1}^1 \frac{\partial v_j^t(k-1+p|k)}{\partial u^t(k-\tau+1+r+1|k)}$$

4. Simulation results

4.1. Benchmark neutralization process

The process under consideration is a neutralization reactor (pH reactor) [7]. This process is a well known and frequently used benchmark for comparing different models and control algorithms, e.g. the Hammerstein and Wiener structures are used to model the neutralization process in [6, 15], identification of the process using the Takagi-Sugeno fuzzy model is considered in [3], identification using a dynamic back propagation algorithm is described in [10]. Fuzzy adaptive control applied to the neutralization process is discussed in [13], MPC algorithms based on neural Wiener models are described in [15].

The neutralization reactor is shown schematically in Fig. 3. In the tank acid HNO_3, base NaOH and buffer $NaHCO_3$ are mixed. pH value of the product is controlled by manipulating the flowrate q_1 of acid. From the modeling point of view the process has one input (q_1) and one output (pH). The fundamental model of the process consists of differential equations [7]

$$\frac{\mathrm{d}W_a(t)}{\mathrm{d}t} = \left(\frac{W_{a_1} - W_a(t)}{V}\right) q_1(t) + \left(\frac{W_{a_2} - W_a(t)}{V}\right) q_2 + \left(\frac{W_{a_3} - W_a(t)}{V}\right) q_3$$

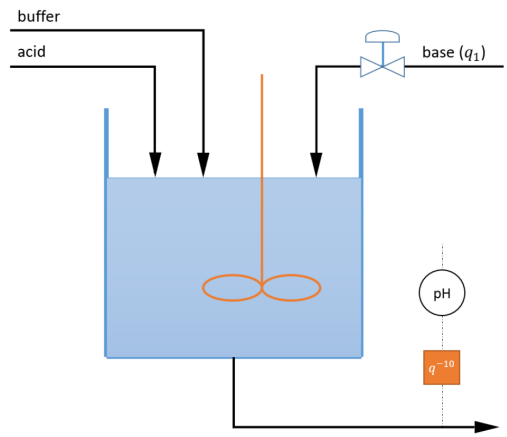

Figure 3. Neutralization reactor with delayed pH measurement

$$\frac{\mathrm{d}W_b(t)}{\mathrm{d}t} = \left(\frac{W_{b_1} - W_b(t)}{V}\right) q_1(t) + \left(\frac{W_{b_2} - W_b(t)}{V}\right) q_2 + \left(\frac{W_{b_3} - W_b(t)}{V}\right) q_3$$

and the algebraic equation

$$W_a(t) + 10^{pH(t)-14} - 10^{-pH(t)} + W_b(t)\frac{1 + 2 \times 10^{pH(t)-pK_2}}{1 + 10^{pK_1 - pH(t)} + 10^{pH(t)-pK2}} = 0 \qquad (23)$$

The sampling time used is 10 seconds. Unlike the cited works concerned with modeling and control of the neutralization process, in this study it is assumed the value of pH is measured with a significant delay equal to 10 discrete sampling instants (i.e. 100 seconds).

The nominal operating point and the parameters of the fundamental model of the reactor are given in Table 2. The discussed neutralization reactor is highly non-linear. In particular, its static characteristics shown in Fig. 4 is nonlinear.

4.2. Modeling of the neutralization process for MPC

Because the model next used in MPC should have the ability to mimic the real process, it is necessary to use the model capable of data generalization, i.e. for different operating conditions the model output must be similar to that of the real process. In order to do so, for identification of the neural networks three independent data sets (generated from the open-loop simulations of the fundamental model) are used: the training data set (Fig. 5), the validation data set (Fig. 6) and the test data set (Fig. 7). The first one is used only for model training, i.e. the model errors (5) is minimized for this set. Although the consecutive iterations of the model training (optimization) algorithm always lead to minimization of the model error, in order to have good generalization and eliminate overtraining the model error for the validation data set (the validation error) is calculated

Table 2. Operating point and the parameters of the fundamental model of the reactor

The nominal operating point of the reactor	
$q_{1_0} = 15.55$ ml/s	$pH_0 = 7.0$
$W_{a_0} = -4.32 \times 10^{-4}$ mol	$W_{b_0} = 5.28 \times 10^{-4}$ mol
Parameters of the fundamental model	
$q_2 = 0.55$ ml/s	$q_3 = 16.60$ ml/s
$pK_1 = 6.35$	$pK_2 = 10.25$
$W_{a_1} = -3.05 \times 10^{-3}$ mol	$W_{b_1} = 5 \times 10^{-5}$ mol
$W_{a_2} = -3 \times 10^{-2}$ mol	$W_{b_2} = 3 \times 10^{-2}$ mol
$W_{a_3} = 3 \times 10^{-3}$ mol	$W_{b_3} = 0$ mol
$V = 2900$ ml	

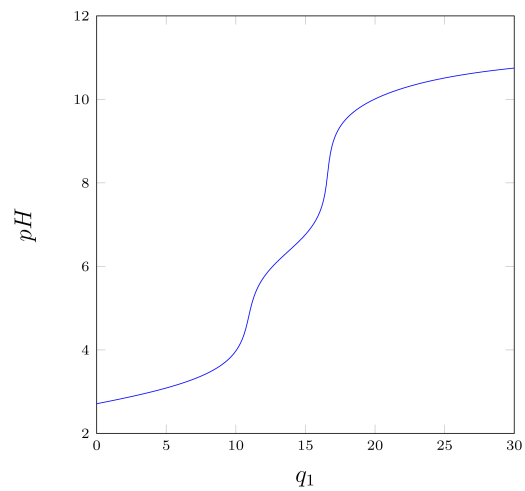

Figure 4. Static characteristic of the neutralization reactor

at each iteration of the training algorithm. Training is finished when the validation error increases, because that is a sign that the model begins to lose its ability to generalize data. Further learning is likely to give a model with too strict dependence on the training data set only. The validation data set is also used for model comparison and selection, the third set is used for final assessment of the chosen model. In order to have data sets representative enough that the learned neural network could properly model all possible operating conditions of the real process, all three sets consist of 1200 samples, as the input signal a series of random step changes is used. In order to use inputs of the networks of a similar order of magnitude, the process variables are scaled: $u = (q_1 - q_{1_0})/15$ and

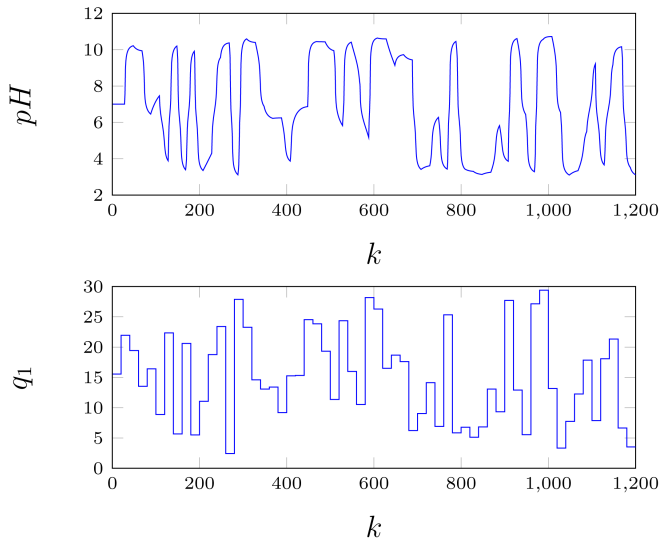

Figure 5. The training data set

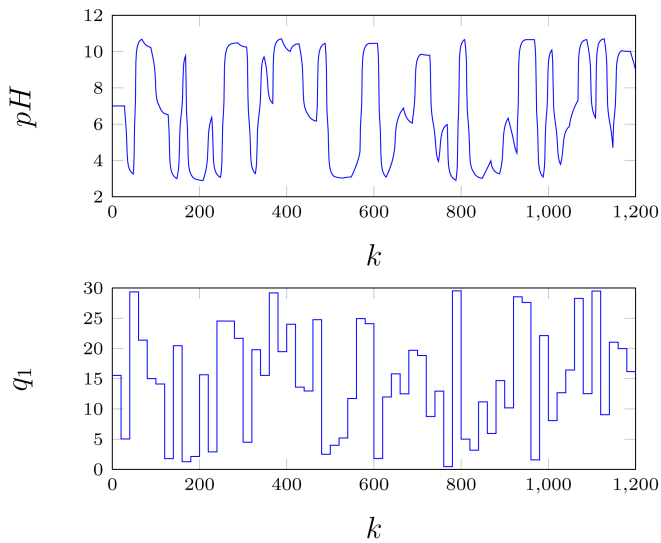

Figure 6. The validation data set

$y = (pH - pH_0)/4$, where q_{1_0} and pH_0 denote the nominal operating point specified in Table 2.

The selection of the best network architecture, i.e. the number of hidden nodes, is done by training a set of models and comparing their complexity and accuracy (errors).

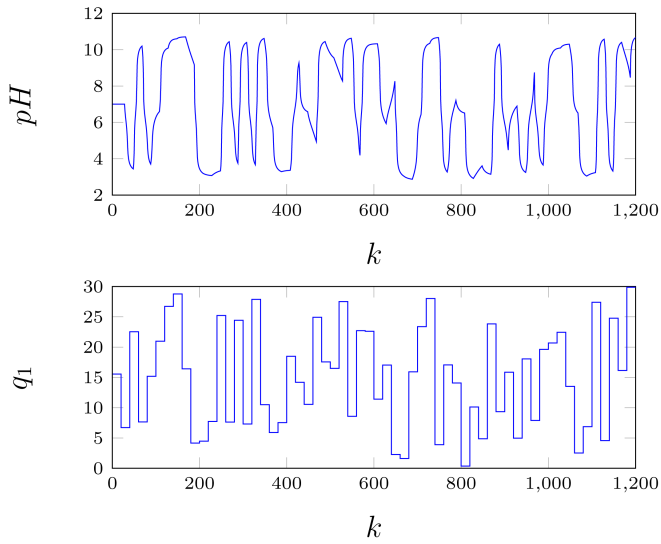

Figure 7. The test data set

In each case (i.e. for each model configuration) training is repeated 10 times and the best results are presented. The classical Elman neural structure with 5, 10, 15, 16, 17, 18 and 19 hidden nodes are considered. Table 3 presents for each model structure the training and validation errors and the number of model parameters. In the case of the modified Elman neural structure the networks with 1, 2, 3, 4, 5, 6 and 7 hidden nodes are considered. Their errors and the number of parameters are compared in Table 4. It is observed that the classical Elman neural network needs many hidden nodes (and weights), the errors of the networks with 5 and 10 nodes are comparable with those of the modified networks containing only 1 or 2 nodes. The classical Elman network with many hidden nodes must be used whereas the modified structure needs only a few nodes. Considering the validation error, the classical Elman network with 17 hidden neurons is chosen, because the models with a lower or higher number of hidden nodes give worse accuracy. In the case of the modified Elman neural network, the structure with 5 hidden neurons is chosen, because for that network the validation error has its lowest value and the error for the test data set is small enough. It is noteworthy that the modified Elman network with 3 neurons has a similar validation error to that of the classical Elman structure with 17 neurons, but the error for the test data set is bigger, so it is a better idea to choose the modified Elman structure since it has a lower number of parameters than the classical one. It is necessary to point out that although the classical network with 17 hidden nodes has bigger validation and test errors than those of the modified one, the first network has as many as 341 weights and the second one has only 41 weights. Fig. 8 compares the validation data set and the output of two models: the classical network with $K = 17$ nodes and the modified one with $K = 5$ nodes are considered. Although numerical values of

their errors, i.e. for the whole validation set (Tables 3 and 4) are similar, for some samples the classical network seems to give unwanted approximation of the process output signal. That observation is true when one considers enlarged fragments of the comparison of the validation data set and model outputs shown in Fig. 9. Correlation between the outputs of the two considered models and the validation data set is depicted in Fig. 10. Finally, it is interesting to compare the classical and modified networks with only $K = 5$ hidden nodes. Fig. 11 compares the validation data set and model outputs whereas Fig. 12 depicts the correlation between the model output and the validation data set. Such a comparison demonstrates the great advantage of the modified Elman network, i.e. for the same model complexity it is much more precise than the classical Elman structure.

Table 3. Comparison of the number of parameters (NP) and accuracy of *the classical recurrent Elman neural network* ($E_{\mathrm{tr}}(\boldsymbol{w})$ – error for the training data set, $E_{\mathrm{ver}}(\boldsymbol{w})$ – error for the validation data set, $E_{\mathrm{test}}(\boldsymbol{w})$ – error for the test data set)

Model	NP	$E_{\mathrm{tr}}(\boldsymbol{w})$	$E_{\mathrm{ver}}(\boldsymbol{w})$	$E_{\mathrm{test}}(\boldsymbol{w})$
$K = 5$	41	2.9804×10^{-1}	2.6214	–
$K = 10$	131	7.3093×10^{-2}	1.9317	–
$K = 15$	271	3.3227×10^{-2}	1.1643	–
$K = 16$	305	3.9353×10^{-3}	9.3754×10^{-1}	–
$K = 17$	341	3.1126×10^{-3}	9.1780×10^{-1}	9.5342×10^{-1}
$K = 18$	379	2.1590×10^{-3}	9.4753×10^{-1}	–
$K = 19$	419	1.5484×10^{-3}	9.4631×10^{-1}	–

Table 4. Comparison of the number of parameters (NP) and accuracy of *the modified recurrent Elman neural network* ($E_{\mathrm{tr}}(\boldsymbol{w})$ – error for the training data set, $E_{\mathrm{ver}}(\boldsymbol{w})$ – error for the validation data set, $E_{\mathrm{test}}(\boldsymbol{w})$ – error for the test data set)

Model	NP	$E_{\mathrm{tr}}(\boldsymbol{w})$	$E_{\mathrm{ver}}(\boldsymbol{w})$	$E_{\mathrm{test}}(\boldsymbol{w})$
$K = 1$	5	1.3369×10^{-2}	2.3022	–
$K = 2$	11	7.7507×10^{-3}	1.7598	–
$K = 3$	19	5.2788×10^{-3}	9.1853×10^{-1}	1.6912
$K = 4$	29	3.9320×10^{-3}	8.8345×10^{-1}	1.1790
$K = 5$	41	1.0173×10^{-3}	6.1099×10^{-1}	7.3211×10^{-1}
$K = 6$	55	9.6266×10^{-4}	7.4933×10^{-1}	–
$K = 7$	71	8.3091×10^{-4}	7.2398×10^{-1}	–

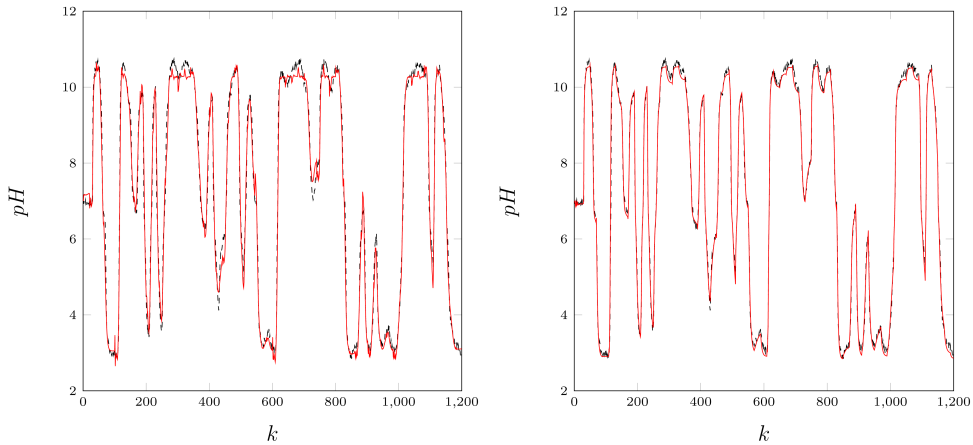

Figure 8. Comparison of the validation data set (*dashed line*) vs. the model output (*solid line*): the classical Elman recurrent neural network with $K = 17$ nodes (*left*), the modified Elman recurrent neural network with $K = 5$ nodes (*right*)

5. Predictive control of the process

5.1. MPC with nonlinear optimization (MPC-NO) based on classical and modified Elman neural networks

At first the two considered Elman structures of a similar validation error, i.e. the classical structure with as many as $K = 17$ hidden nodes and the modified structure with only $K = 5$ nodes, are compared in the MPC algorithm in which the full nonlinear model is used for prediction and a nonlinear optimization problem (9) is solved at each sampling instant. If a model is precise enough, such an approach to MPC gives the best possible control performance, the MPC-NO algorithm may be regarded as the "ideal" one. Simulation results are depicted in Fig. 13. Each simulation is performed under the same conditions, the same values of parameters of the algorithm: prediction horizon ($N = 20$), control horizon ($N_u = 2$) and $\lambda = 1$. Although both models have similar modeling errors (Tables 3 and 4), but the classical network is less capable of modeling the delayed process (Fig. 9). A direct consequence of this fact is that in the case of the MPC-NO algorithm based on the classical Elman network there are some vanishing oscillations in the manipulated and controlled variables. Conversely, in the case of the MPC-NO algorithm based on the modified Elman network, the output trajectory quickly follows the set-point trajectory.

5.2. Efficiency of the MPC-NPLPT algorithms based on the modified Elman network

There are three parameters for tuning quality of MPC-NPLPT algorithm, i.e. δ_u, δ_y and t_{max}. They determine the number of internal iterations of the algorithm. The lower

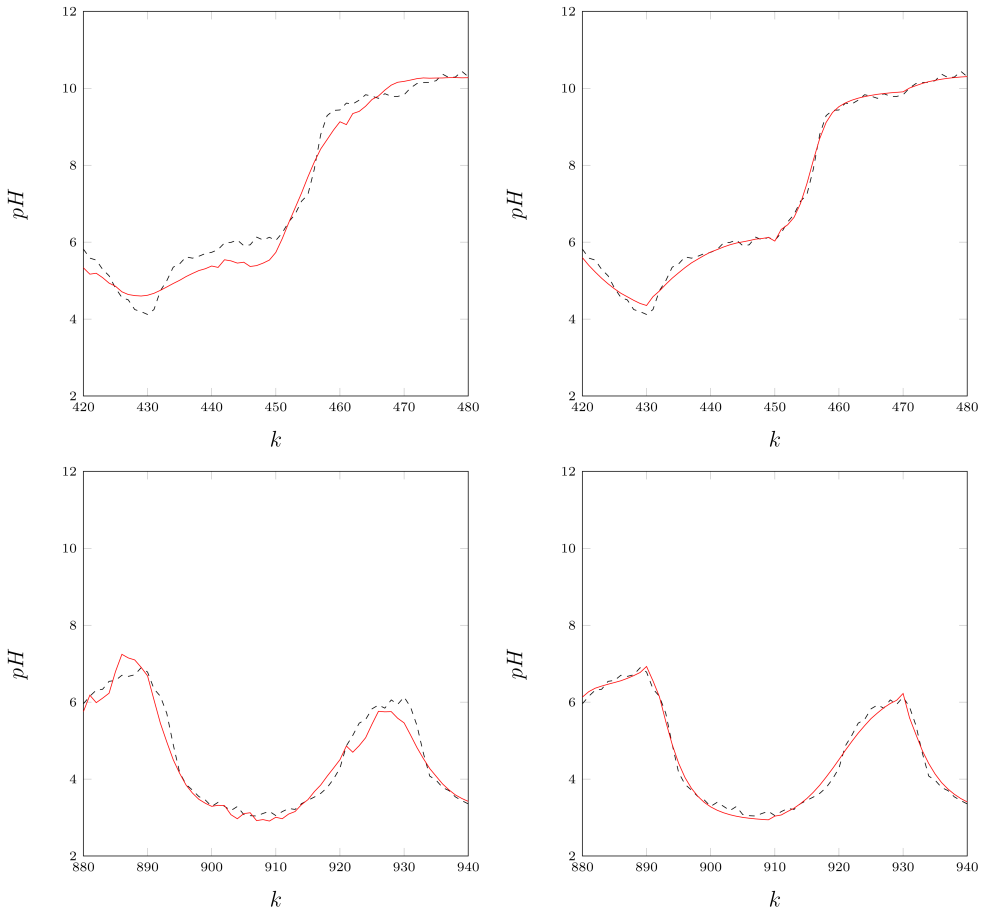

Figure 9. Comparison of two fragments of the validation data set (*dashed line*) vs. the model output (*solid line*) for $k = 420, \ldots, 480$ and $k = 880, \ldots, 940$: the classical Elman recurrent neural network with $K = 17$ nodes (*left*), the modified Elman recurrent neural network with $K = 5$ nodes (*right*)

the values of δ_u and δ_y, the more internal iterations are necessary and the more internal iterations, the better the control quality.

Simulation performance of the MPC-NPLPT algorithm with successive on-linearization and quadratic optimization is evaluated. Fig. 14 compares the trajectories obtained in MPC-NO and MPC-NPLPT algorithms. Both algorithms use the same modified Elman neural network and parameters $\delta_u = 10^{-9}$, $\delta_y = 5 \times 10^{-8}$, $t_{\max} = 9$. The comparison clearly indicates that the MPC-NPLPT control strategy gives practically the same trajectories as the "ideal" MPC-NO approach with on-line nonlinear optimization.

Fig. 15 compares the trajectories obtained in MPC-NO and MPC-NPLPT algorithms with additional constraints imposed on the rate of change of the manipulated variable,

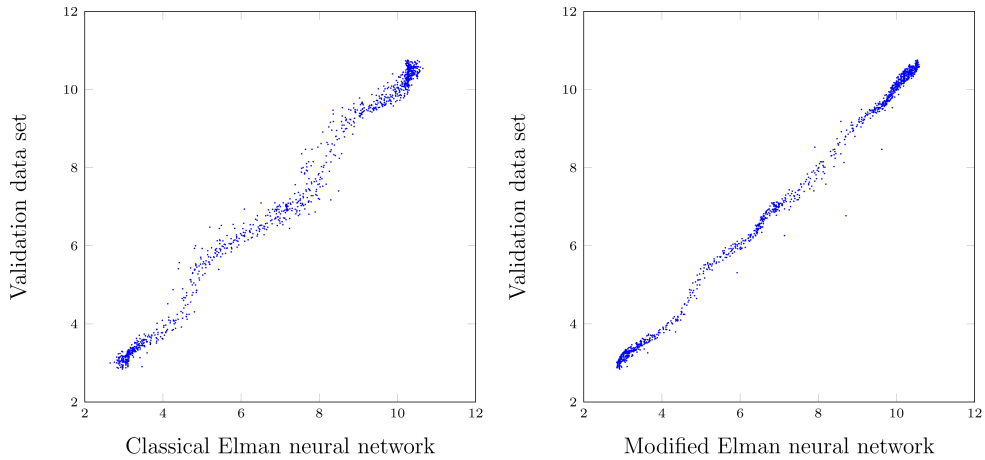

Figure 10. The correlation between the model output and the validation data set: the classical Elman recurrent neural network with $K = 17$ nodes (*left*), the modified Elman recurrent neural network with $K = 5$ nodes (*right*)

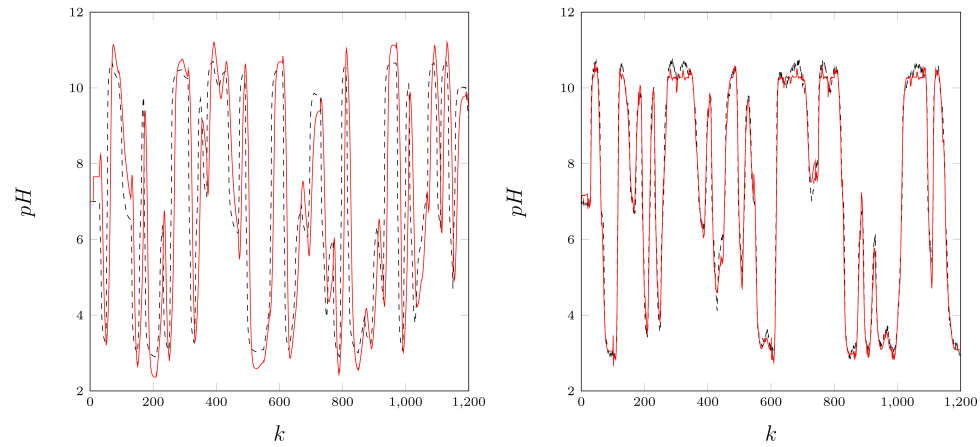

Figure 11. Comparison of the validation data set (*dashed line*) vs. the model output (*solid line*): the classical Elman recurrent neural network with $K = 5$ nodes (*left*), the modified Elman recurrent neural network with $K = 5$ nodes (*right*)

$\triangle u^{max} = 1$. Due to the constraints, the obtained trajectories are slower in comparison with those shown in Fig. 14.

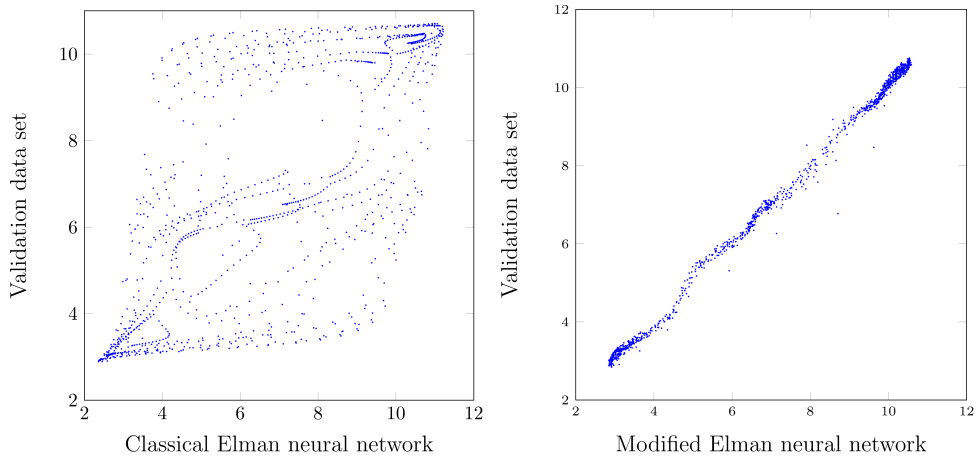

Figure 12. The correlation between the model output and the validation data set: the classical Elman recurrent neural network with $K = 5$ nodes (*left*), the modified Elman recurrent neural network with $K = 5$ nodes (*right*)

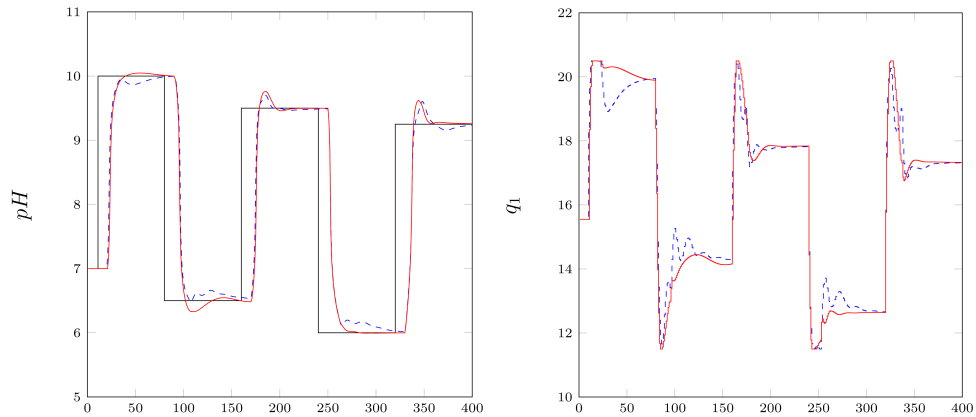

Figure 13. Simulation results of the MPC-NO algorithm based on the classical Elman neural network (*dashed line*) and based on the modified Elman neural network (*solid line*)

6. Summary

The paper describes a modified recurrent neural network of the Elman type. The modification consists in taking into account the process delay. It is observed that for significantly delayed systems the classical network needs a huge number of parameters whereas the modified network requires only a portion of the weights to provide a similar modeling accuracy. For the considered polymerization reactor the classical network has as many as 17 hidden nodes and 341 parameters whereas the modified structure of a similar accuracy has only 5 nodes and 41 weights. Furthermore, it is also observed that

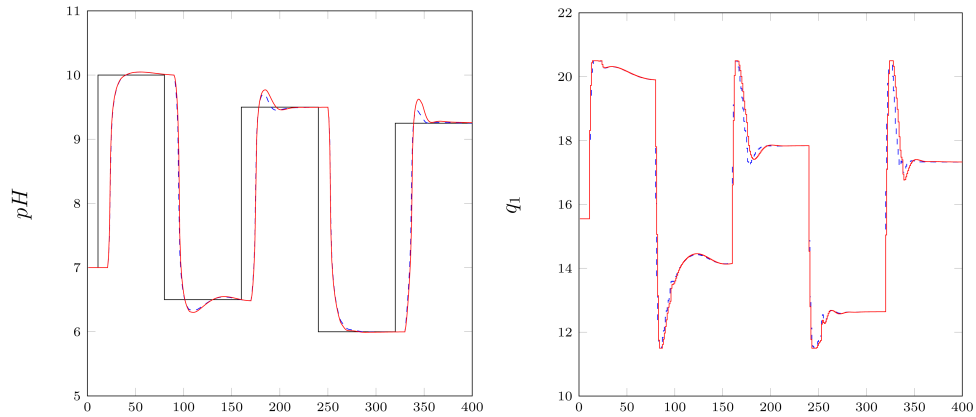

Figure 14. Simulation results of the MPC-NO algorithm based on the modified Elman neural network (*dashed line*) and the MPC-NPLPT algorithm based on the same network (*solid line*)

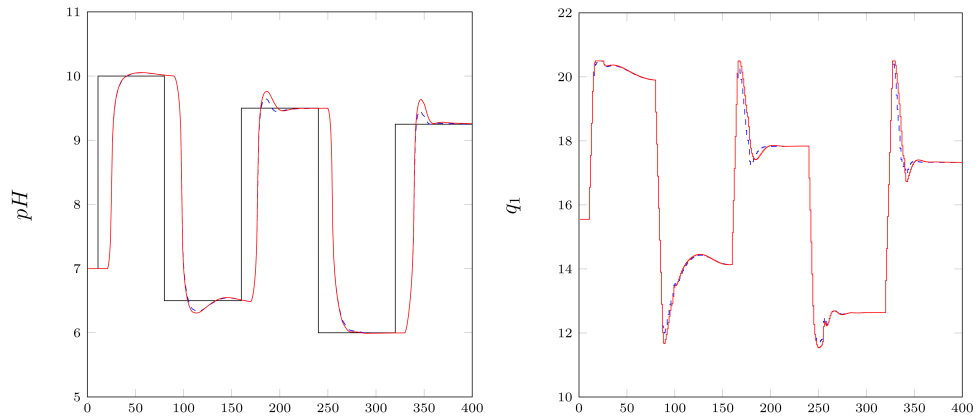

Figure 15. Simulation results of the MPC-NO algorithm based on the modified Elman neural network (*dashed line*) and the MPC-NPLPT algorithm based on the same network (*solid line*) with additional constraints imposed on the rate of change of the manipulated variable

the modified network approximates the trajectory of delayed systems in a more natural way. The introduced modification of the neural network may be also used in the case of different recurrent neural structures, e.g. in the Real Time Recurrent Network (RTRN) [11]. Furthermore, it is straightforward to use the discussed modification in Elman and RTRN networks which are used for modeling of multiple-input multiple-output dynamic systems.

This paper also describes the application of the modified Elman neural network in the MPC algorithm with successive on-line linearization of the predicted trajectory. Trajectory linearization makes it possible to obtain a simple quadratic optimization MPC task. For the considered nonlinear neutralization process, the trajectories obtained in the

algorithm with on-line linearization are very similar to those possible in the MPC approach with repetitive nonlinear optimization.

References

[1] R. BONNEAU, M. T. FACCIOTTI, D. J. REISS, A. K. SCHMID, M. PAN, A. KAUR, V. THORSSON, P. SHANNON, M. H. JOHNSON, J. C. BARE, W. LONGABAUGH, M. VUTHOORI, K. WHITEHEAD, A. MADAR, L. SUZUKI, T. MORI, D. E CHANG, J. DIRUGGIERO, C. H. JOHNSON, L. HOOD and N. S. BALIGA: A predictive model for transcriptional control of physiology in a free living cell. *Cell*, **131**(7), (2007), 1354-1365.

[2] E. F. CAMACHO and C. BORDONS: Model Predictive Control. Springer, London, 1999.

[3] X. CHEN, J. CHEN and B. LEI: Identification of pH neutralization process based on the T-S fuzzy model, **215** CCIS of Communications in Computer and Information Science. Springer, Berlin, Heidelberg, 2011.

[4] F. DECLERCQ and R. DE KEYSER: Comparative study of neural predictors in model based predictive control. In *Proc. of Int. Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing*, (1996), 20-28.

[5] J. L. ELMAN: Finding structure in time. *Cognitive Science*, **14**(2), (1990), 179-211.

[6] J. C. GÓMEZ and E. BAEYENS: Subspace-based identification algorithms for hammerstein and wiener models. *European J. of Control*, **11**(2), (2005), 127-136.

[7] J. C. GÓMEZ, A. JUTAN and E. BAEYENS: Wiener model identification and predictive control of a ph neutralisation process. *IEE Proceedings: Control Theory and Applications*, **151**(3), (2004), 329-338.

[8] K. A. GREENE, K. W. BAUER JR., M. KABRISKY, S. K. ROGERS and G. F. WILSON: Estimating pilot workload using Elman recurrent neural networks: a preliminary investigation. In *Intelligent Engineering Systems Through Artificial Neural Networks*, **7** (1997), 703-708.

[9] M. T. HAGAN, H. B. DEMUTH, MARK BEALE and O. DE JESUS: Neural Network Design. PWS Publishing Co, Boston, MA, USA, 1996.

[10] B. HAN and M. HAN: Nonlinear time delay systems identification based on dynamic bp algorithm. *Dalian Ligong Daxue Xuebao/Journal of Dalian University of Technology*, **50**(5), (2010), 777-781.

[11] S. HAYKIN: Neural Networks: A Comprehensive Foundation. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.

[12] R. HOVORKA, V. CANONICO, L. J. CHASSIN, U. HAUETER, M. MASSI-BENEDETTI, M. O. FEDERICI, T. R. PIEBER, H. C. SCHALLER, L. SCHAUPP, T. VERING and M. E. WILINSKA: Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes. *Physiological Measurement*, **25**(4), (2004), 905-920.

[13] Y. JIA, L. Y ZHANG and T. Y CHAI: Based on fuzzy adaptive control of model predictive in slurry neutralization process. *Dongbei Daxue Xuebao/Journal of Northeastern University*, **35**(5), (2014), 617-621.

[14] M. ŁAWRYŃCZUK: Accuracy and computational efficiency of suboptimal nonlinear predictive control based on neural models. *Applied Soft Computing J.*, **11**(2), (2011), 2202-2215.

[15] M. ŁAWRYŃCZUK: Practical nonlinear predictive control algorithms for neural wiener models. *J. of Process Control*, **23**(5), (2013), 696-714.

[16] M. ŁAWRYŃCZUK: Computationally Efficient Model Predictive Control Algorithms. Springer, 2014.

[17] P. LI, Y. LI, Q. XIONG, Y. CHAI and Y. ZHANG: Application of a hybrid quantized Elman neural network in short-term load forecasting. Int. J. of Electrical Power and Energy Systems, **55** (2014), 749-759.

[18] Q. LI, Y. QIN, Z.Y. WANG, Z.X. ZHAO, M.H. ZHAN and Y. LIU: Prediction of urban rail transit sectional passenger flow based on elman neural network. *Applied Mechanics and Materials*, **505-506** (2014), 1023-1027.

[19] J. W LIOU, W. C CHENG, J. C HUANG and C. Y LIOU: Distributed representation of word by using Elman network. *Int. J. of Intelligent Information and Database Systems*, **7**(4), (2013), 373-386.

[20] J. M. MACIEJOWSKI: Predictive Control with Constraints. Prentice Hall, Harlow, 2002.

[21] D. P. MANDIC and J. CHAMBERS: Recurrent Neural Networks for Prediction: Learning Algorithms,Architectures and Stability. John Wiley & Sons, Inc, New York, NY, USA, 2001.

[22] A. MARCINIAK and J. KORBICZ: Modular Neural Networks pages 135-177. Biocybernetyka i inżynieria biomedyczna 2000. **6** Sieci neuronowe, ISBN: 83-87674-18-4. Akademicka Oficyna Wydaw. EXIT, Warszawa, 2000, (in Polish).

[23] J. M. ZAMARRE NO and P. VEGA: State-space neural network. properties and application. *Neural Networks*, **11**(6), (1998), 1099-1112.

[24] J. NOCEDAL and S.J. WRIGHT: Numerical Optimization. Springer, Berlin, New York, 2006.

[25] S. OSOWSKI: Neural Networks for Information Processing. Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa, 2006, (in Polish).

[26] P. PLAWIAK and R. TADEUSIEWICZ: Approximation of phenol concentration using novel hybrid computational intelligence methods. *Int. J. of Applied Mathematics and Computer Science*, **24**(1), (2014), 165-181.

[27] S. J. QIN and T. BADGWELL: A survey of industrial model predictive control technology. *Control Engineering Practice*, **11**(7), (2003), 733-764.

[28] D. SAMEK: Elman neural networks in model predictive control. In *Proc. of the 23rd European Conf. on Modelling and Simulation*, (2009), 577-581.

[29] S. SEKER, E. AYAZ and E. TÜRKCAN: Elman's recurrent neural network applications to condition monitoring in nuclear power plant and rotating machinery. *Engineering Applications of Artificial Intelligence*, **16**(7-8), (2003), 647-656.

[30] P. TATJEWSKI: *Advanced Control of Industrial Processes, Structures and Algorithms*. Springer, London, 2007.

[31] J. G WU and H. LUNDSTEDT: Prediction of geomagnetic storms from solar wind data using Elman recurrent neural networks. *Geophysical Research Letters*, **23**(4), (1996), 319-322.

[32] C. XIA, X. XIANG, B. LEI and D. PENG: Research on a dynamic recurrent Elman neural network model for electric load forecasting and its wilcoxon test. *J. of Computational Information Systems*, **6**(3), (2010), 959-966.

[33] J. XU and M. ZHANG: Neural network modeling and generalized predictive control for an autonomous underwater vehicle. In *6th IEEE Int. Conf. on Industrial Informatics*, (2008), 487-491.

[34] C. ZHOU, L. Y. DING and R. HE: Pso-based Elman neural network model for predictive control of air chamber pressure in slurry shield tunneling under yangtze river. *Automation in Construction*, **36** (2013), 208-217.