Neurocomputing 205 (2016) 311–328

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Modelling and predictive control of a neutralisation reactor using sparse support vector machine Wiener models

Maciej Ławryńczuk*

Institute of Control and Computation Engineering, Warsaw University of Technology, ul. Nowowiejska 15/19, 00–665 Warsaw, Poland

ARTICLE INFO

Article history:
Received 19 November 2015
Received in revised form
5 February 2016
Accepted 17 March 2016
Communicated by Ngoc Thanh Nguyen
Available online 10 May 2016

Keywords:
Process control
Neutralisation reactor control
Model Predictive Control
Wiener models
Least-Squares Support Vector Machines

ABSTRACT

This paper has two objectives: (a) it describes the problem of finding a precise and uncomplicated model of a neutralisation process, (b) it details development of a nonlinear Model Predictive Control (MPC) algorithm for the plant. The model has a cascade Wiener structure, i.e. a linear dynamic part is followed by a nonlinear steady-state one. A Least-Squares Support Vector Machine (LS-SVM) approximator is used as the steady-state part. Although the LS-SVM has excellent approximation abilities and it may be found easily, it suffers from a huge number of parameters. Two pruning methods of the LS-SVM Wiener model are described and compared with a classical pruning algorithm. The described pruning methods make it possible to remove as much as 70% of support vectors without any significant deterioration of model accuracy. Next, the pruned model is used in a computationally efficient MPC algorithm in which a linear approximation of the predicted output trajectory is successively found on-line and used for prediction. The control profile is calculated on-line from a quadratic optimisation problem. It is demonstrated that the described MPC algorithm with on-line linearisation based on the pruned LS-SVM Wiener model gives practically the same trajectories as those obtained in the computationally complex MPC approach based on the full model with on-line nonlinear optimisation repeated at each sampling instant.

1. Introduction

Good control of neutralisation processes is necessary in chemical engineering, biotechnology and waste-water treatment industries [16]. Both steady-state and dynamic properties of the neutralisation process are nonlinear, which means that it is difficult to control by the classical linear control methods (e.g. PID), in particular when the set-point or other operating conditions change significantly and fast. In addition to its industrial significance, the neutralisation process is a classical benchmark used for evaluation of different nonlinear model structures and control methods. Due to nonlinearity of the process adaptive control techniques may be used, in particular a model reference adaptive neural network control strategy [23], an adaptive nonlinear output feedback control scheme containing an input–output linearising controller and a nonlinear observer [15], an adaptive nonlinear Internal Model Controller (IMC) [22] and an adaptive backstepping state feedback controller [47]. An alternative is to use multi-model controllers, e.g. a multi-model PID controller based on a set of simple linear dynamic models [5], a multi-model robust H_∞ controller [11], or fuzzy structures, e.g. a fuzzy PI controller [10], a fuzzy PID controller [19] and a fuzzy IMC structure [21]. An adaptive fuzzy sliding mode controller is presented in [7], a nonlinear IMC structure is discussed in [30]. Another options are a neural network linearising scheme cooperating with a PID controller [23], a model-free learning controller using reinforcement learning [41] and an approximate multi-parametric nonlinear MPC controller [14].

Unlike the classical control approaches, such as PID, in which the model of the process is used only during development of the controller, in Model Predictive Control (MPC) algorithms [42] a dynamic model of the controlled process is used directly on-line. The model calculates predictions of the output (or state) variables, which are next used during optimisation of the control sequence. Prediction and optimisation are repeatedly performed on-line. Optimisation makes it possible not only to find the best possible control profile which results in excellent set-point tracking and disturbance compensation, but also to take into account constraints imposed on process inputs (manipulated variables) and outputs (controlled variables) or state variables in a natural and efficient manner. Furthermore, the MPC approach is very universal as it allows to control multiple-input multiple-output processes. That is why the MPC algorithms have been successfully used for years in numerous advanced applications. Example applications of MPC include a mobile robot [1], an automotive engine [2], an active queue management system in TCP/IP networks [4], a flexible

* Tel.: +48 22 234 71 24; fax: +48 22 825 37 19.
E-mail address: M.Lawrynczuk@ia.pw.edu.pl

http://dx.doi.org/10.1016/j.neucom.2016.03.066
0925-2312/

manipulator [9], an electric vehicle [27], a cutting process [33], a multi-tank water system [36], an unmanned aerial vehicle [38], a distillation column [43], an air conditioning system [46].

The neutralisation process may be controlled by MPC algorithms. A multiple-model control strategy based on a set of classical linear MPC controllers is described in [8,11]. A neural network trained off-line to mimic the nonlinear MPC algorithm may be also used [3]. A continuous-time MPC algorithm using a piecewise-linear approximation, which simplifies implementation, is discussed in [31]. When a nonlinear model is used directly in MPC for prediction, the MPC optimisation problem solved at each sampling instant on-line is a nonlinear task. Applications of such nonlinear MPC algorithms to the neutralisation process are reported in [26,45]. On-line nonlinear optimisation is not only computationally demanding, but also convergence problems are possible, the obtained solution may be a local minimum, not the global ones. A practical solution leading to reduction of computational burden is to use for prediction not the full nonlinear model, but its local linear approximation or an approximation of the predicted trajectory [24]. Successive on-line linearisation makes it possible to eliminate the necessity of solving on-line a nonlinear MPC optimisation problem as at each sampling instant an easy to solve quadratic optimisation task is solved. Nonlinear MPC algorithms with successive on-line model or trajectory linearisation applied to the neutralisation process are described in [25]. An excellent review of possible MPC approaches to the neutralisation process is given in [16].

The basic issue to address during development of MPC algorithms is the choice of the model. Although a number of black-box model structures exist [28], e.g. polynomials, fuzzy systems, neural networks, wavelets, etc., in case of the neutralisation reactor a cascade block-oriented Wiener model may be efficiently used, e.g. [23,26,30,45]. The Wiener structure consists of a linear dynamic part and a nonlinear steady-state one connected in series [12,18]. As the steady-state part of the Wiener model a neural network may be used [25]. Neural networks are excellent approximators, but training (although performed off-line) is a quite demanding nonlinear optimisation problem. In order to find a good neural model a number of networks (with different initial weights and different number of hidden nodes) are trained and the best one is finally chosen for application. An interesting alternative is to use a Support Vector Machine (SVM) approximator [37]. Although the SVM model is nonlinear, its identification requires solving convex optimisation problems, typically quadratic programming ones. An extension of the SVM approximator is a Least Squares Support Vector Machine (LS-SVM), whose identification is even simpler as only least-squares problems are solved [39]. An important disadvantage of LS-SVM is lack of spareness, i.e. the number of support vectors is the same as the number of training samples. To reduce the number of parameters some pruning algorithms may be used, e.g. the approach discussed by the authors of the LS-SVM approximator [40] which consists in eliminating the support vectors with the smallest absolute value of spectrum. A more complicated pruning algorithm is detailed in [20], the sequential minimal optimisation (SMO) pruning method is introduced in [48].

This paper reports model identification and pruning of the neutralisation reactor. Two pruning methods of the Wiener LS-SVM model are compared with a classical pruning algorithm. Next, the MPC algorithm with successive on-line trajectory linearisation and quadratic optimisation is developed for the pruned model of the process. The discussed MPC algorithm is compared with a computationally complex MPC approach with on-line nonlinear optimisation repeated at each sampling instant. The effect of model pruning on its quality, control performance of MPC and its computational complexity is discussed. Although both SVM and LS-SVM approximators have been used in MPC, e.g. a multiple-tank system process is considered in [17], a flight control problem in [38] and an air-conditioning system in [46], in the cited works computationally demanding on-line nonlinear optimisation is used at each sampling instant.

This paper is organised as follows. Section 2 reminds the general idea of MPC and Section 3 describes the structure of the LS-SVM Wiener model. The main parts of the paper, given in Sections 4 and 5, discuss the MPC algorithm with on-line trajectory linearisation for the LS-SVM Wiener model, its identification and pruning. Section 6 thoroughly discusses development of the model and predictive control of the considered neutralisation reactor. Finally, Section 7 concludes the paper.

2. Predictive control problem formulation

Let the input (the manipulated variable) of the considered dynamic system be denoted by u and the output (the controlled output) of the system be denoted by y. In MPC algorithms [42] at each consecutive sampling instant k not only the current value $u(k)$ of the manipulated variable is calculated, but a set of future increments

$$\Delta \boldsymbol{u}(k) = \left[\Delta u(k|k)\ \Delta u(k+1|k) \ldots \Delta u(k+N_u-1|k)\right]^T \tag{1}$$

is found, where N_u is the control horizon and the increments are defined as

$$\Delta u(k+p|k) = \begin{cases} u(k|k)-u(k-1) & \text{if } p=0 \\ u(k+p|k)-u(k+p-1|k) & \text{if } p \geq 1 \end{cases}$$

The symbol $u(k+p|k)$ denotes the value of the input signal for the future sampling instant $k+p$ calculated at the current instant k. It is assumed that $\Delta u(k+p|k)=0$ for $p \geq N_u$. The objective of the MPC algorithm is to minimise differences between the set-point trajectory and the corresponding predicted values of the output signal over the prediction horizon, $N \geq N_u$, and to penalise excessive control increments. Hence, the future decision variables of MPC (Eq. (1)) are determined from an optimisation procedure. The cost-function is typically

$$J(k) = \sum_{p=1}^{N} \left(y^{sp}(k+p|k) - \hat{y}(k+p|k)\right)^2 + \sum_{p=0}^{N_u-1} \lambda \left(\Delta u(k+p|k)\right)^2 \tag{2}$$

where the set-point for the sampling instant $k+p$ known at the current instant k is $y^{sp}(k+p|k)$ (very frequently it is assumed that $y^{sp}(k+p|k)=y^{sp}(k)$ for all $p=1,\ldots,N$), the future value of the process output signal predicted for the instant $k+p$ at the instant k is denoted by $\hat{y}(k+p|k)$, $\lambda>0$ is a weighting coefficient (the bigger the λ, the slower the algorithm). The problem of tuning MPC algorithms, i.e. adjusting parameters λ, N, N_u, is discussed elsewhere [42]. If it is necessary to take into account some constraints imposed on the manipulated and controlled variables, the future control increments (1) are found on-line at each sampling instant from the following optimisation problem

$$\begin{aligned} &\min_{\Delta u(k|k),\ldots,\Delta u(k+N_u-1|k)} \left\{J(k)\right\} \\ &\text{subject to} \quad u^{min} \leq u(k+p|k) \leq u^{max}, \quad p=0,\ldots,N_u-1 \\ &-\Delta u^{max} \leq \Delta u(k+p|k) \leq \Delta u^{max}, \quad p=0,\ldots,N_u-1 \\ &y^{min} \leq \hat{y}(k+p|k) \leq y^{max}, \quad p=1,\ldots,N \end{aligned} \tag{3}$$

where u^{min}, u^{max}, Δu^{max}, y^{min}, y^{max} define constraints imposed on the magnitude of the input variable, on the increment of the input variable and on the magnitude of the predicted output variable, respectively. The MPC optimisation task (3) is solved on-line at each sampling instant which gives the future control increments (1), but only the first element of the determined sequence is applied to the process, i.e. $u(k)=\Delta u(k|k)+u(k-1)$. At the next

sampling instant, $k+1$, the prediction is shifted one step forward and the whole procedure is repeated.

3. Wiener model with LS-SVM approximator

Predicted values of the output variable, $\hat{y}(k+p|k)$, over the prediction horizon (i.e. for $p=1,\ldots,N$) are calculated from a dynamic model of the process. In this work for modelling the cascade Wiener model [12,18] depicted in Fig. 1 is used. It consists of a linear dynamic part connected in series with nonlinear steady-state part, $v(k)$ denotes an auxiliary signal between two blocks of the model. The linear part of the Wiener model is described by the equation

$$\boldsymbol{A}(q^{-1})v(k)=\boldsymbol{B}(q^{-1})u(k)$$

where the polynomials are

$$\boldsymbol{A}(q^{-1})=1+a_1q^{-1}+\cdots+a_{n_A}q^{-n_A}$$
$$\boldsymbol{B}(q^{-1})=b_\tau q^{-\tau}+\cdots+b_{n_B}q^{-n_B}$$

The backward shift operator is denoted by q^{-1}, the integers n_A, n_B, τ define the order of dynamics, $\tau\leq n_B$, the constant parameters of the linear dynamic part are denoted by the real numbers a_j ($j=1,\ldots,n_A$) and b_j ($j=\tau,\ldots,n_B$). The output of the linear part of the model is

$$v(k)=\sum_{j=\tau}^{n_B}b_ju(k-j)-\sum_{j=1}^{n_A}a_jv(k-j) \tag{4}$$

The steady-state part of the model is described by the general equation

$$y(k)=g(v(k))$$

where the differentiable function $g:\mathbb{R}\rightarrow\mathbb{R}$ is realised by the LS-SVM approximator [39]. It has one input, n_{sv} support vectors and one output. Assuming the exponential kernel function, its output is

$$y(k)=\beta+\sum_{i=1}^{n_{sv}}\alpha_i\exp\left(-\frac{(v(k)-v_{sv,i})^2}{\sigma^2}\right) \tag{5}$$

where the quantities $v_{sv,i}$ define support vectors (since the model has one input, the quantities $v_{sv,i}$ are scalars), α_i and β are model parameters determined during training, σ is a parameter of the kernel. The output of the LS-SVM Wiener model can be explicitly expressed as a function of the input signal of the process and the auxiliary signal of the model at some previous sampling instant taking into account Eqs. (4) and (5), which gives

$$y(k)=\beta+\sum_{i=1}^{n_{sv}}\alpha_i\exp\left(-\sigma^{-2}\left(\sum_{j=\tau}^{n_B}b_ju(k-j)-\sum_{j=1}^{n_A}a_jv(k-j)-v_{sv,i}\right)^2\right) \tag{6}$$

4. MPC based on Wiener models with LS-SVM approximator

In MPC algorithms an explicit dynamic model is used in order to predict future behaviour of the process, i.e. to calculate the predicted values of the output variable, $\hat{y}(k+p|k)$, over the prediction horizon, i.e. for $p=1,\ldots,N$. The general prediction equation is

$$\hat{y}(k+p|k)=y(k+p|k)+d(k) \tag{7}$$

where the quantities $\hat{y}(k+p|k)$ are the predicted output values whereas the model output signal for the future sampling instant $k+p$ calculated at the current instant k is $y(k+p|k)$. In order to compensate for the disturbances which affect the process and the unavoidable mismatch between the process and its model, an estimation of the unmeasured disturbances $d(k)$ is used in Eq. (7). The disturbance is assessed as the difference between the measured value of the output signal and a corresponding value calculated from the model using measurements up to the previous sampling instant $(k-1)$, i.e.

$$d(k)=y(k)-y(k|k-1) \tag{8}$$

For the considered Wiener model with LS-SVM approximator, from Eqs. (6) and (8) one has

$$d(k)=y(k)-\beta-\sum_{i=1}^{n_{sv}}\alpha_i\exp\left(-\sigma^{-2}\left(\sum_{j=\tau}^{n_B}b_ju(k-j)-\sum_{j=1}^{n_A}a_jv(k-j)-v_{sv,i}\right)^2\right) \tag{9}$$

From Eqs. (5) and (7) it is possible to calculate the predictions of the output variable over the whole prediction horizon

$$\hat{y}(k+p|k)=\beta+\sum_{i=1}^{n_{sv}}\alpha_i\exp\left(-\sigma^{-2}(v(k+p|k)-v_{sv,i})^2\right)+d(k) \tag{10}$$

where the prediction of the auxiliary model variable v for the sampling instant $k+p$ at the current instant k is calculated using Eq. (4) from

$$v(k+p|k)=\sum_{j=1}^{I_{uf}(p)}b_ju(k-\tau+1-j+p|k)+\sum_{j=I_{uf}(p)+1}^{I_u}b_ju(k-\tau+1-j+p)-\sum_{j=1}^{I_{yp}(p)}a_jv(k-j+p|k)-\sum_{j=I_{yp}(p)+1}^{n_A}a_jv(k-j+p) \tag{11}$$

Some additional integer numbers are $I_{uf}(p)=\max(\min(p-\tau+1,I_u),0)$, $I_u=n_B-\tau+1$, $I_{yp}(p)=\min(p-1,n_A)$. It is straightforward to notice that due to a nonlinear nature of the LS-SVM Wiener model (more specifically, to its steady-state part), the future predictions of the process output variable, $\hat{y}(k+p|k)$, are nonlinear functions of the repeatedly calculated on-line decision variables of the MPC algorithm, i.e. the future control increments (1). A direct consequence of this fact is that the MPC optimisation problem (3) becomes a nonlinear task which must be solved at each sampling instant on-line. Such an approach is referred to in this paper as the MPC algorithm with Nonlinear Optimisation (MPC-NO). Its apparent disadvantage is the necessity of using a nonlinear solver and significant computational burden of on-line calculations.

In this paper the MPC algorithm with Nonlinear Prediction and Linearisation along the Predicted Trajectory (MPC-NPLPT) for the Wiener system with the LS-SVM approximator is developed. A detailed description of the algorithm, but for a different model structure, is given in [24,25]. In short, at each sampling instant of the MPC-NPLPT algorithm a linear approximation of the predicted output trajectory is calculated on-line. As the approximated trajectory is a linear function of the calculated future control increments, it is possible to formulate an easy to solve quadratic optimisation problem, nonlinear optimisation is not necessary. For good approximation accuracy (which results in good control performance), trajectory linearisation and optimisation of the control increments are repeated a few times at each sampling instant in internal iterations. The predicted nonlinear output trajectory

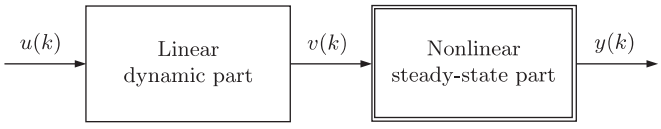

Fig. 1. The structure of the LS-SVM Wiener model.

vector, defined for the prediction horizon N, in the internal iteration t is

$$\hat{\boldsymbol{y}}^t(k)=\left[\hat{y}^t(k+1|k)\ \ldots\ \hat{y}^t(k+N|k)\right]^{\mathrm{T}} \tag{12}$$

The predicted trajectory is linearised along some future control trajectory, defined over the control horizon, N_{u}, found in the previous internal iteration $(t-1)$

$$\boldsymbol{u}^{t-1}(k)=\left[u^{t-1}(k|k)\ \ldots\ u^{t-1}(k+N_{\mathrm{u}}-1|k)\right]^{\mathrm{T}} \tag{13}$$

Using Taylor's series expansion method, a linear approximation of the nonlinear output trajectory $\hat{\boldsymbol{y}}^t(k)$ along the input trajectory $\boldsymbol{u}^{t-1}(k)$, i.e. linearisation of the function $\hat{\boldsymbol{y}}^t(\boldsymbol{u}^t(k)):\mathbb{R}^{N_{\mathrm{u}}}\rightarrow\mathbb{R}^N$ is

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

The predicted output trajectory corresponding to the input trajectory $\boldsymbol{u}^{t-1}(k)$, which is calculated at the previous internal iteration, is denoted by $\hat{\boldsymbol{y}}^{t-1}(k)$. The future values of the process input signal calculated at the current internal iteration t are denoted by $\boldsymbol{u}^t(k)$. The vectors $\hat{\boldsymbol{y}}^{t-1}(k)$ and $\hat{\boldsymbol{u}}^t(k)$ are similar to the vectors $\hat{\boldsymbol{y}}^t(k)$ and $\boldsymbol{u}^{t-1}(k)$ (Eqs. (12) and (13)), respectively, i.e. $\hat{\boldsymbol{y}}^{t-1}(k)=\left[\hat{y}^{t-1}(k+1|k)\ \ldots\ \hat{y}^{t-1}(k+N|k)\right]^{\mathrm{T}}$, $\boldsymbol{u}^t(k)=\left[u^t(k|k)\ \ldots\ u^t(k+N_{\mathrm{u}}-1|k)\right]^{\mathrm{T}}$. The matrix of the derivatives of the predicted output trajectory with respect of the future control signals is of dimensionality $N\times N_{\mathrm{u}}$ and it has the structure

$$\boldsymbol{H}^t(k)=\left.\frac{\mathrm{d}\hat{\boldsymbol{y}}(k)}{\mathrm{d}\boldsymbol{u}(k)}\right|_{\substack{\hat{\boldsymbol{y}}(k)=\hat{\boldsymbol{y}}^{t-1}(k)\\ \boldsymbol{u}(k)=\boldsymbol{u}^{t-1}(k)}}=\frac{\mathrm{d}\hat{\boldsymbol{y}}^{t-1}(k)}{\mathrm{d}\boldsymbol{u}^{t-1}(k)}$$
$$=\begin{bmatrix}\dfrac{\partial\hat{y}^{t-1}(k+1|k)}{\partial u^{t-1}(k|k)} & \cdots & \dfrac{\partial\hat{y}^{t-1}(k+1|k)}{\partial u^{t-1}(k+N_{\mathrm{u}}-1|k)}\\ \vdots & \ddots & \vdots\\ \dfrac{\partial\hat{y}^{t-1}(k+N|k)}{\partial u^{t-1}(k|k)} & \cdots & \dfrac{\partial\hat{y}^{t-1}(k+N|k)}{\partial u^{t-1}(k+N_{\mathrm{u}}-1|k)}\end{bmatrix} \tag{15}$$

The elements of the nonlinear predicted trajectory $\hat{\boldsymbol{y}}^{t-1}(k)$ are calculated from the prediction equation corresponding to the general MPC prediction method defined by Eq. (7), but the current internal iteration of the MPC-NPLPT algorithm must be taken into account, i.e. for $p=1,\ldots,N$ one has

$$\hat{y}^{t-1}(k+p|k)=y^{t-1}(k+p|k)+d(k) \tag{16}$$

where the model output signal calculated at the current instant k for the future sampling instant $k+p$ and for the future control trajectory $\boldsymbol{u}^{t-1}(k)$ is $y^{t-1}(k+p|k)$. The disturbance estimation is found from Eq. (9). The elements of the nonlinear output trajectory predicted at the current sampling instant k and the internal iteration $t-1$ for the instant $k+p$ are calculated using Eqs. (5) and (16), similar to Eq. (10), which gives

$$\hat{y}^{t-1}(k+p|k)=\beta+\sum_{i=1}^{n_{\mathrm{sv}}}\alpha_i\exp\left(-\sigma^{-2}(v^{t-1}(k+p|k)-v_{\mathrm{sv},i})^2\right)+d(k) \tag{17}$$

where from Eq. (4), similar to Eq. (11) the prediction of the auxiliary model variable v at the current instant k for the internal iteration $t-1$ and the sampling instant $k+p$ is calculated using Eq. (4), which gives

$$v^{t-1}(k+p|k)=\sum_{j=1}^{l_{\mathrm{uf}}(p)}b_j u^{t-1}(k-\tau+1-j+p|k)+\sum_{j=l_{\mathrm{uf}}(p)+1}^{l_{\mathrm{u}}}b_j u(k-\tau+1-j+p)-\sum_{j=1}^{l_{\mathrm{yp}}(p)}a_j v^{t-1}(k-j+p|k)-\sum_{j=l_{\mathrm{yp}}(p)+1}^{n_{\mathrm{A}}}a_j v(k-j+p) \tag{18}$$

The entries of the matrix (15), i.e. the derivatives of the predicted output trajectory for the previous internal iteration with respect to the future control scenario from the previous internal iteration, are calculated by differentiating Eqs. (17), which lead to

$$\frac{\partial y^{t-1}(k+p|k)}{\partial u^{t-1}(k+r|k)}=2\sigma^{-2}\sum_{i=1}^{n_{\mathrm{sv}}}\alpha_i\exp\left(-\sigma^{-2}(v^{t-1}(k+p|k)-v_{\mathrm{sv},i})^2\right)\times(v_{\mathrm{sv},i}-v^{t-1}(k+p|k))\frac{\partial v^{t-1}(k+p|k)}{\partial u^{t-1}(k+r|k)} \tag{19}$$

The derivatives of the predicted value of the auxiliary signal v with respect to the future control trajectory are calculated differentiating Eq. (18) for all $p=1,\ldots,N$, $r=0,\ldots,N_{\mathrm{u}}-1$, which gives

$$\frac{\partial v^{t-1}(k+p|k)}{\partial u^{t-1}(k+r|k)}=\sum_{j=1}^{l_{\mathrm{uf}}(p)}b_j\frac{\partial u^{t-1}(k-\tau+1-j+p|k)}{\partial u^{t-1}(k+r|k)}-\sum_{j=1}^{l_{\mathrm{yp}}(p)}a_j\frac{\partial v^{t-1}(k-j+p|k)}{\partial u^{t-1}(k+r|k)} \tag{20}$$

Because $u(k+p|k)=u(k+N_{\mathrm{u}}-1|k)$ for $p\geq N_{\mathrm{u}}$, the first partial derivatives on the right side of Eq. (20) are

$$\frac{\partial u^{t-1}(k+p|k)}{\partial u^{t-1}(k+r|k)}=\begin{cases}1 & \text{if } p=r, p>r \text{ and } r=N_{\mathrm{u}}-1\\ 0 & \text{otherwise}\end{cases} \tag{21}$$

Because $\boldsymbol{u}^t(k)=\boldsymbol{J}\Delta\boldsymbol{u}^t(k)+\boldsymbol{u}(k-1)$ where the vector $\boldsymbol{u}(k-1)=\left[u(k-1)\ \ldots\ u(k-1)\right]^{\mathrm{T}}$ is of length N_{u} and the matrix $\boldsymbol{J}$ of dimensionality $N_{\mathrm{u}}\times N_{\mathrm{u}}$ has the following structure

$$\boldsymbol{J}=\begin{bmatrix}1 & 0 & 0 & \ldots & 0\\ 1 & 1 & 0 & \ldots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & 1 & 1 & \ldots & 1\end{bmatrix}$$

the linear approximation of the nonlinear predicted output trajectory (14) becomes

$$\hat{\boldsymbol{y}}^t(k)=\boldsymbol{H}^t(k)\boldsymbol{J}\Delta\boldsymbol{u}^t(k)+\hat{\boldsymbol{y}}^{t-1}(k)+\boldsymbol{H}^t(k)(\boldsymbol{u}(k-1)-\boldsymbol{u}^{t-1}(k)) \tag{22}$$

Due to linearisation the predicted output trajectory calculated for the current sampling instant k and at the current internal iteration t, i.e. $\hat{\boldsymbol{y}}^t(k)$, is a linear function of the future control increments $\Delta\boldsymbol{u}^t(k)$. Using the prediction equation (22), the general MPC optimisation problem (3) becomes the following quadratic programming task

$$\min_{\Delta\boldsymbol{u}^t(k)}\ \left\{\|\boldsymbol{y}^{\mathrm{sp}}(k)-\boldsymbol{H}^t(k)\boldsymbol{J}\Delta\boldsymbol{u}^t(k)-\hat{\boldsymbol{y}}^{t-1}(k)-\boldsymbol{H}^t(k)(\boldsymbol{u}(k-1)-\boldsymbol{u}^{t-1}(k))\|^2+\left\|\Delta\boldsymbol{u}^t(k)\right\|^2_{\boldsymbol{\Lambda}}\right\}$$
$$\text{subject to}\quad \boldsymbol{u}^{\min}\leq\boldsymbol{J}\Delta\boldsymbol{u}^t(k)+\boldsymbol{u}(k-1)\leq\boldsymbol{u}^{\max}$$
$$-\Delta\boldsymbol{u}^{\max}\leq\Delta\boldsymbol{u}^t(k)\leq\Delta\boldsymbol{u}^{\max}$$
$$\boldsymbol{y}^{\min}\leq\boldsymbol{H}^t(k)\boldsymbol{J}\Delta\boldsymbol{u}^t(k)+\hat{\boldsymbol{y}}^{t-1}(k)+\boldsymbol{H}^t(k)(\boldsymbol{u}(k-1)-\boldsymbol{u}^{t-1}(k))\leq\boldsymbol{y}^{\max} \tag{23}$$

where the norm of a vector $\boldsymbol{x}$ is defined as $\|\boldsymbol{x}\|^2_{\boldsymbol{A}}=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}$, the set-point trajectory

$$\boldsymbol{y}^{\mathrm{sp}}(k)=\left[y^{\mathrm{sp}}(k+1|k)\ \ldots\ y^{\mathrm{sp}}(k+N|k)\right]^{\mathrm{T}}$$

is the vector of length N, the input constraint vectors $\boldsymbol{u}^{\min}=\left[u^{\min}\ \ldots\ u^{\min}\right]^{\mathrm{T}}$, $\boldsymbol{u}^{\max}=[u^{\max}\ \ldots\ u^{\max}]^{\mathrm{T}}$, $\Delta\boldsymbol{u}^{\max}=\left[\Delta u^{\max}\ \ldots\ \Delta u^{\max}\right]^{\mathrm{T}}$ are of length N_{u}, the output constraint vectors $\boldsymbol{y}^{\min}=\left[y^{\min}\ \ldots\ y^{\min}\right]^{\mathrm{T}}$, $\boldsymbol{y}^{\max}=[y^{\max}\ \ldots\ y^{\max}]^{\mathrm{T}}$ are of length N, the weighting matrix $\boldsymbol{\Lambda}=\mathrm{diag}(\lambda,\ldots,\lambda)$ is of dimensionality $N_{\mathrm{u}}\times N_{\mathrm{u}}$.

At each sampling instant k of the MPC-NPLPT algorithm for the dynamic system described by the Wiener system with the LS-SVM approximator the following steps are repeated:

1. The LS-SVM Wiener model is used to assess the unmeasured disturbance $d(k)$ from Eq. (9).
2. The first internal iteration ($t=1$): the LS-SVM Wiener model is used to find from Eqs. (16) to (18) the future output trajectory $\hat{\boldsymbol{y}}^0(k)$ which corresponds to the initial input trajectory $\boldsymbol{u}^0(k)$. The initial input trajectory may be initialised using the last N_u-1 elements of the optimal control sequence calculated at the previous sampling instant $(k-1)$.
3. A linear approximation (22) of the output predicted trajectory $\hat{\boldsymbol{y}}^1(k)$ along the input trajectory $\boldsymbol{u}^0(k)$ is found using the LS-SVM Wiener model, i.e. the entries of the matrix $\boldsymbol{H}^1(k)$ defined by Eq. (15) are obtained from Eqs. (19) to (21).
4. The MPC-NPLPT quadratic optimisation problem (23) is solved to find the future control increments $\Delta\boldsymbol{u}^1(k)$ in the first internal iteration.
5. If the process is far from the desired set-point, i.e. when

$$\sum_{p=0}^{N_0}\left\|y^{sp}(k-p)-y(k-p)\right\|^2 \geq \delta_y \tag{24}$$

the internal iterations are continued for $t=2,\ldots,t_{max}$.

5.1. The LS-SVM Wiener model is used to calculate the predicted output trajectory $\hat{\boldsymbol{y}}^{t-1}(k)$ corresponding to the future input trajectory $\boldsymbol{u}^{t-1}(k)=\boldsymbol{J}\Delta\boldsymbol{u}^{t-1}(k)+\boldsymbol{u}(k-1)$ from Eqs. (16) to (18).

5.2. A linear approximation (22) of the predicted output trajectory $\hat{\boldsymbol{y}}^t(k)$ along the input trajectory $\boldsymbol{u}^{t-1}(k)$ is found using the LS-SVM Wiener model, i.e. the entries of the matrix $\boldsymbol{H}^t(k)$ defined by Eq. (15) are obtained from Eqs. (19) to (21).

5.3. The MPC-NPLPT quadratic optimisation problem (23) is solved to find the future control increments $\Delta\boldsymbol{u}^t(k)$ in the current internal iteration t.

5.4. If the difference between future control increments calculated in two consecutive internal iterations is small, i.e. when

$$\left\|\Delta\boldsymbol{u}^t(k)-\Delta\boldsymbol{u}^{t-1}(k)\right\|^2 < \delta_u \tag{25}$$

or $t>t_{max}$, internal iterations are terminated. Otherwise, the internal iteration index is increased ($t:=t+1$), the algorithm goes to step 5.1.

6. The first element of the determined sequence $\Delta\boldsymbol{u}^t(k)$ is applied to the process, i.e. $u(k)=\Delta u^t(k|k)+u(k-1)$.
7. At the next sampling instant ($k:=k+1$) the algorithm starts from step 1.

In addition to N, N_u and λ, the tuning parameters of the MPC-NPLPT algorithm are N_0, δ_u and δ_y.

5. Identification and pruning of Wiener models with LS-SVM approximator

5.1. Model identification

It is assumed that the nonlinear LS-SVM steady-state part of the Wiener model is determined using the given set of training steady-state samples denoted by $\{v_i,y_i\}_{i=1,\ldots,n_s}$, where v is its input and y is the output. The objective is to find the relation between the input and the output as an analytical function. Considering the classical LS-SVM approximator [39], the following relation is used in the so-called primal weight space

$$y_i(v_i)=w^T\varphi(v_i)+\beta,\quad i=1,\ldots,n_s \tag{26}$$

where $\varphi(\cdot)$ is a function which maps the input space into the so-called higher dimensional (possibly infinite dimensional) feature space, the weight vector w is of the same dimension as the feature space, the bias is β. The approximation error for the training sample i is defined as the difference between the data sample and the model output, i.e. $e_i=y_i-y_i^{mod}$. In order to find the parameters of the approximator, the following optimisation problem in primal weight space is considered

$$\min_{w,\beta}\left\{J(w,e)=\frac{1}{2}w^Tw+\frac{1}{2}\gamma\sum_{i=1}^{n_s}e_i^2\right\}$$
$$\text{subject to}\quad y_i=w^T\varphi(v_i)+\beta+e_i,\quad i=1,\ldots,n_s \tag{27}$$

The cost function J consists of the classical sum of squared model errors ($\sum_{i=1}^{n_s}e_i^2$) and a regularisation term w^Tw, which is a frequent technique used in model training, e.g. neural networks. The relative importance of these two terms is determined by the positive real constant γ. In general, the weight vector w can be infinite-dimensional, which makes solution of the optimisation problem (27) impossible. That is why the model is calculated in the dual space rather than in the primal space. For this purpose the Lagrangian is defined as

$$\mathcal{L}(w,\beta,e,\alpha)=J(w,e)-\sum_{i=1}^{n_s}\alpha_i(w^T\varphi(v_i)+\beta+e_i-y_i)$$

where the Lagrange multipliers α_i are called support values. Taking into account the optimality conditions (it is necessary to equal the derivatives of the function $\mathcal{L}$ to zero), one may easily find that

$$\begin{bmatrix}0 & \mathbf{1}^T\\ \mathbf{1} & \Omega+\gamma^{-1}\boldsymbol{I}\end{bmatrix}\begin{bmatrix}\beta\\ \alpha\end{bmatrix}=\begin{bmatrix}0\\ y\end{bmatrix} \tag{28}$$

where $\mathbf{1}=[1\ \ldots\ 1]^T$, $\alpha=[\alpha_1\ \ldots\ \alpha_{n_s}]^T$, $y=[y_1\ \ldots\ y_{n_s}]^T$ are vectors of length n_s, $\boldsymbol{I}$ is an identity matrix of dimensionality $n_s\times n_s$ and the entries of the matrix Ω are determined as $\Omega_{k,l}=\varphi^T(v_k)\varphi(v_l)$ for $k,l=1,\ldots,n_s$. Taking into account the Mercer's condition [39], the resulting LS-SVM model is

$$y(v)=\sum_{i=1}^{n_s}\alpha_iK(v,v_i)+\beta \tag{29}$$

where model parameters α_i, for $i=1,\ldots,n_s$, and β are obtained from solving the set of linear equation (28). In this work the RBF kernel is used, i.e. $K(v_k,v_l)=\exp(-\|v_k-v_l\|/\sigma^2)$. Because the input of the approximator, v, is a scalar, one obtains the LS-SVM model

$$y=\beta+\sum_{i=1}^{n_{sv}}\alpha_i\exp\left(-\frac{(v-v_{sv,i})^2}{\sigma^2}\right) \tag{30}$$

where the number of support vectors is equal to the number of samples, i.e. $n_{sv}=n_s$. The model (30) is next used in the nonlinear steady-state part of the Wiener model (Eq. (5)). For the calculated LS-SVM steady-state part of the Wiener model, its linear dynamic part is found using the prediction error method, i.e. the sum of squared differences between model output and the training data is minimised for a given dynamic training data set.

5.2. Model pruning

Although the LS-SVM approximator has two important advantages, i.e. easiness of training (it is only necessary to solve the set of linear equation (28)) and very good approximation accuracy, it also has an important disadvantage as the number of support vectors is exactly the same as the number of training patterns. Therefore, it is desirable to prune the LS-SVM approximator to remove the least important model parameters. In this study three pruning methods are considered. In the first two approaches to pruning it is assumed that the nonlinear steady-state part of the LS-SVM Wiener model is found and pruned independently using the steady-state training and validation data sets. Next, the linear dynamic part of the model is found using the dynamic training and validation data sets for the

obtained steady-state part. In the last approach the steady-state data is still used for training the LS-SVM approximator, but accuracy of the whole LS-SVM Wiener model is assessed using the dynamic data.

The pruning algorithm 1 is discussed in [40]. In short, it may be summarised as follows:

1. The LS-SVM approximator is trained using n_s steady-state data points.
2. A small amount of points (e.g. 5% of the set) with the smallest values in the sorted spectrum $|\alpha_i|$ is removed from the training set.
3. The LS-SVM approximator is retrained using the reduced steady-state training set.
4. The algorithm goes to step 2, unless the user-defined performance index degrades considerably. If the performance becomes worse, it is suggested to recalculate the approximator for some modified values of γ and σ.

During training of the LS-SVM approximator the minimised cost-function consists of two parts: the classical sum of squared errors and a regularisation term (as in the optimisation problem (27)). The performance index used in step 4 to assess the model is the classical sum of squared errors of the steady-state part of the Wiener model

$$\mathrm{SSE_s} = \sum_{i=1}^{n_s} (y_i - y_i^{\mathrm{mod}}(v_i))^2 \tag{31}$$

where y_i denotes the training pattern and $y_i^{\mathrm{mod}}(v_i)$ is the model output for the input v_i. It is important to point out that the approximator may be assessed not using the training data set, but also (or only) an alternative data set, i.e. the validation set.

The pruning algorithm 2 of the LS-SVM approximator can be summarised as follows:

1. The LS-SVM approximator is trained using n_s steady-state data points.
2. As many as n_s-2 candidate LS-SVM approximators are trained by removing one training pattern i from the training data set, $i=2,\ldots,n_s-1$. For each of the candidate approximators the $\mathrm{SSE_s}$ errors (Eq. (31)) for the original training (i.e. not reduced) and validation data sets are calculated. The training pattern for which the value of $\mathrm{SSE_s}$ increases in the minimal way (or decreases in the maximal way) (for the training or validation set) is removed from the training data set.
3. The LS-SVM approximator is retrained using the reduced steady-state training set.
4. The algorithm goes to step 2, unless the performance index $\mathrm{SSE_s}$ degrades considerably or the reduced steady-state training set consists of 2 points.

It is necessary to notice that in contrast to pruning algorithm 1, in the second approach to pruning the decision which training patterns should be removed from the training data set is taken not on the basis of the performance index used for training (as in the optimisation task (27)), because for model assessment the regularisation term is not necessary. Conversely, the model is assessed on the basis of the squared sum of errors (31), which directly described model accuracy. In Step 2 of the pruning algorithm 2 it is impossible to remove the first and the last points from the reduced training data set as they define the operation domain (the same protection may be implemented in the pruning algorithm 1).

In the pruning algorithm 3 the steady-state data is still used for training the LS-SVM approximator, but the decision regarding its pruning is made taken into account the error of the whole LS-SVM Wiener model, i.e. the dynamic data is used for this purpose. In such a case the sum of squared errors of the Wiener model is

$$\mathrm{SSE_d} = \sum_{k=1}^{n_{ds}} (y(k) - y^{\mathrm{mod}}(k))^2 \tag{32}$$

where $y(k)$ denotes the training pattern and $y^{\mathrm{mod}}(k)$ is the model output for the sampling instant k, the number of samples in the dynamic data set is n_{ds}. The pruning algorithm 3 can be summarised as follows:

1. The LS-SVM approximator is trained using n_s steady-state data points.
2. As many as n_s-2 candidate LS-SVM approximators are trained by removing one training pattern i from the training data set, $i=2,\ldots,n_s-1$. For each of the candidate approximators the $\mathrm{SSE_d}$ errors (Eq. (32)) for the dynamic training and validation data sets are calculated. The training pattern for which the value of $\mathrm{SSE_d}$ increases in the minimal way (or decreases in the maximal way) (for the training or validation set) is removed from the steady-state training data set.
3. The LS-SVM approximator is retrained using the reduced steady-state training set.
4. The algorithm goes to step 2, unless the performance index $\mathrm{SSE_d}$ degrades considerably or the reduced steady-state training set consists of 2 points.

Optionally, the third step of all discussed algorithms may be followed by identification of the linear dynamic part of the Wiener model for the pruned steady-state part.

6. Simulation results

6.1. Neutralisation process

The structure of the control system of the neutralisation reactor [13] is depicted in Fig. 2. A base (NaOH) stream q_1, a buffer ($NaHCO_3$) stream q_2 and an acid (HNO_3) stream q_3 are mixed in a constant volume tank. The value of pH is controlled by manipulating the base flow rate q_1 (ml/s). The continuous-time fundamental model of the process is composed of two nonlinear ordinary differential equations

$$\frac{\mathrm{d}W_a(t)}{\mathrm{d}t} = \frac{q_1(t)(W_{a_1} - W_a(t))}{V} + \frac{q_2(t)(W_{a_2} - W_a(t))}{V} + \frac{q_3(t)(W_{a_3} - W_a(t))}{V} \tag{33}$$

$$\frac{\mathrm{d}W_b(t)}{\mathrm{d}t} = \frac{q_1(t)(W_{b_1} - W_b(t))}{V} + \frac{q_2(t)(W_{b_2} - W_b(t))}{V} + \frac{q_3(t)(W_{b_3} - W_b(t))}{V} \tag{34}$$

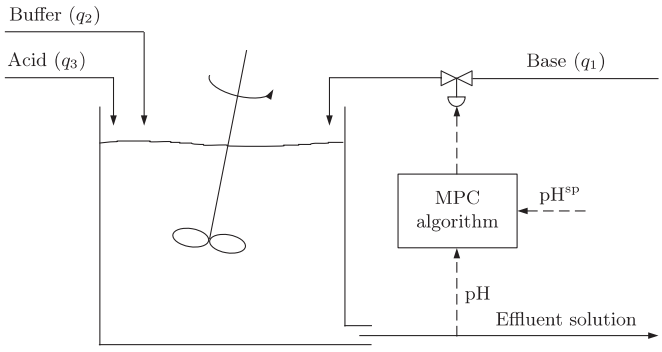

Fig. 2. The structure of the control system of the neutralisation reactor.

and one algebraic output equation

$$W_a(t)+10^{pH(t)-14}-10^{-pH(t)}+W_b(t)\frac{1+2\times 10^{pH(t)-K_2}}{1+10^{K_1-pH(t)}+10^{pH(t)-K_2}}=0 \quad (35)$$

State variables W_a and W_b are reaction invariants. Initial operating conditions of the process are $q_1=15.55$ ml/s, pH=7, $W_a=-4.32\times 10^{-4}$ mol , $W_b=5.28\times 10^{-4}$ mol. The buffer and acid streams are assumed to be constant, $q_2=0.55$ ml/s, $q_3=16.60$ ml/s. All constant parameters of the fundamental model are $K_1=6.35$, $K_2=10.25$, $V=2900$ ml, $W_{a_1}=-3.05\times 10^{-3}$ mol, $W_{a_2}=-3\times 10^{-2}$ mol, $W_{a_3}=3\times 10^{-3}$ mol, $W_{b_1}=5\times 10^{-5}$ mol, $W_{b_2}=3\times 10^{-2}$ mol, $W_{b_3}=0$ mol.

The system of differential equations (33)–(34) is treated as the process during simulations. The equations are solved by means of the Runge–Kutta 45 method. For calculation of the value of pH for simulation purposes, using Eq. (35), a fourth-order polynomial is obtained

$$10^{-14-K_2}x^4(t)+\left(W_a(t)10^{-K_2}+10^{-14}+2W_b(t)10^{-K_2}\right)x^3(t)+\left(W_a(t)+10^{K_1-14}-10^{-K_2}+W_b(t)\right)x^2(t)+\left(W_a10^{K_1}-1\right)x(t)-10^{K_1}=0 \quad (36)$$

where $x(t)=10^{pH(t)}$. Eq. (36) is solved and next its positive solution gives the actual value of pH from $pH(t)=\log_{10}x(t)$. Additionally, it is assumed that the value of pH is measured with a delay equal to 10 s.

The considered neutralisation reactor is a significantly nonlinear system. Firstly, its steady-state characteristics shown in Fig. 3 is nonlinear. It is an interesting fact that the steady-state gain of the process changes from 0.0350 to 2.9416, more than 84 times. Secondly, dynamic properties of the process are also nonlinear. Fig. 4 depicts example step-responses of the process caused by increasing and decreasing the base flow rate q_1 by 1, 5 and 10 ml/s, respectively. Time constants depend on the step value, for positive and negative changes of the manipulated variable the responses have different amplitude and shape. That is why the classical linear control strategies are inefficient for the neutralisation process, e.g. the linear MPC algorithm as discussed in [25].

6.2. Modelling of neutralisation process and model pruning

Taking into account the identification and pruning procedure of the LS-SVM Wiener model described in Section 5, it is necessary to generate steady-state and dynamic data. The dynamic data (random step changes in the input variable q_1 and corresponding responses of the output variable pH) is generated directly from the fundamental model defined by Eqs. (33), (34) and (36). The sampling time (for identification of the dynamic part of the model and for controller design) is 10 s. The dynamic training and validation data sets are depicted in Fig. 5. Both sets have 2000 samples. Additionally, the output signal contains small measurement noise.

From the dynamic fundamental model one may easily obtain the steady-state fundamental description of the process. In the steady-state conditions, from the differential equations (33)–(34), it follows that

$$W_a=\frac{q_1W_{a_1}+q_2W_{a_2}+q_3W_{a_3}}{q_1+q_2+q3} \quad (37)$$

$$W_b=\frac{q_1W_{b_1}+q_2W_{b_2}+q_3W_{b_3}}{q_1+q_2+q3} \quad (38)$$

whereas from the algebraic output equation (36)

$$10^{-14-K_2}x^4+\left(W_a10^{-K_2}+10^{-14}+2W_b10^{-K_2}\right)x^3+\left(W_a+10^{K_1-14}-10^{-K_2}+W_b\right)x^2+\left(W_a10^{K_1}-1\right)x-10^{K_1}=0 \quad (39)$$

and the value of pH is calculated as $pH=\log_{10}x$. From the fundamental stead-state model defined by Eqs. (37)–(39) the steady-state training and validation data sets are generated. They consist of 100 and 70 equidistant data patterns, respectively, in the whole operating domain determined by $q_1=0,\ldots,30$ ml/s.

Both steady-state and dynamic data sets are scaled: $u=q_1-q_{1,0}$, $y=pH-pH_0$, where in the nominal operating point $q_{1,0}=15.5$ and $pH_0=7$.

Before identification, the LS-SVM approximator needs values of two tuning parameters: the regularisation factor γ, which determines the trade-off between the training error minimisation and smoothness (the optimisation task (27)) and the parameter σ of the kernel (Eqs. (5), (30)). After a series of experiments the regularisation parameter is set to $\gamma=1000$ (for small values of γ the approximation suffers from the lack of smoothness). Next, the influence of the kernel parameter σ on the possible approximation accuracy is analysed. For this purpose three LS-SVM approximators of the steady-state nonlinear part of the Wiener model are trained using the complete training data set. The model is not pruned, it contains $n_{sv}=100$ support vectors. Fig. 6 compares the steady-state validation data set and model output, model errors are also given. It may be noticed that the bigger the parameter σ, the bigger the errors, in particular in the central part of the steady-state characteristics. The model error SSE_s (Eqs. (31)) is

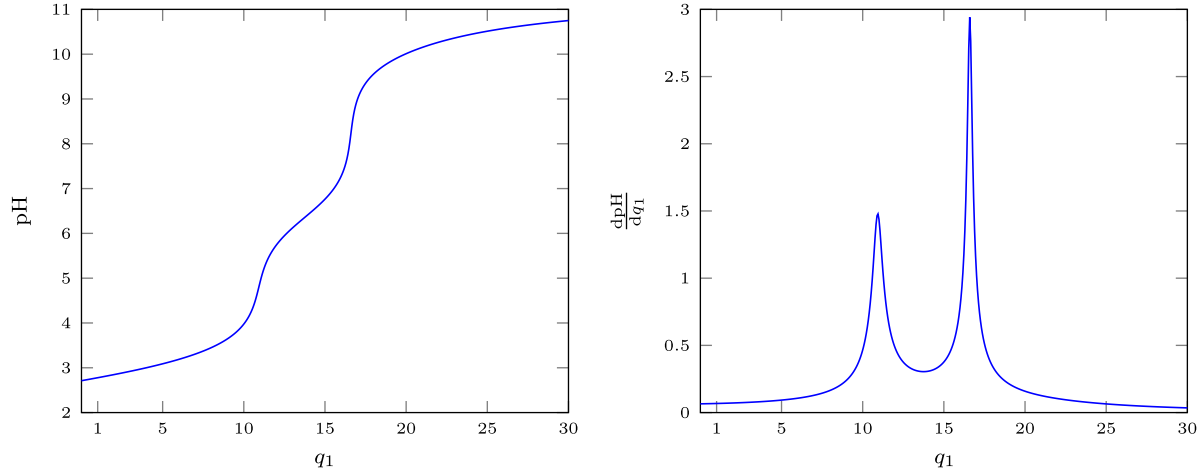

Fig. 3. The steady-state characteristics of the neutralisation reactor (*left panel*) and its steady-state gain (*right panel*).

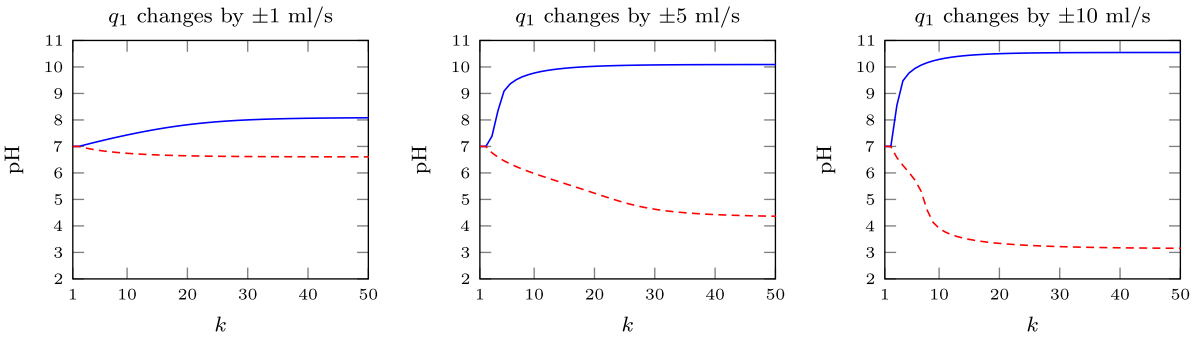

Fig. 4. Step-responses of the reactor (the value of pH) caused by increasing (*solid line*) and decreasing (*dashed line*) the base flow rate q_1 by 1 ml/s (*left panel*), 5 ml/s (*middle panel*) and 10 ml/s (*right panel*) at $k=1$.

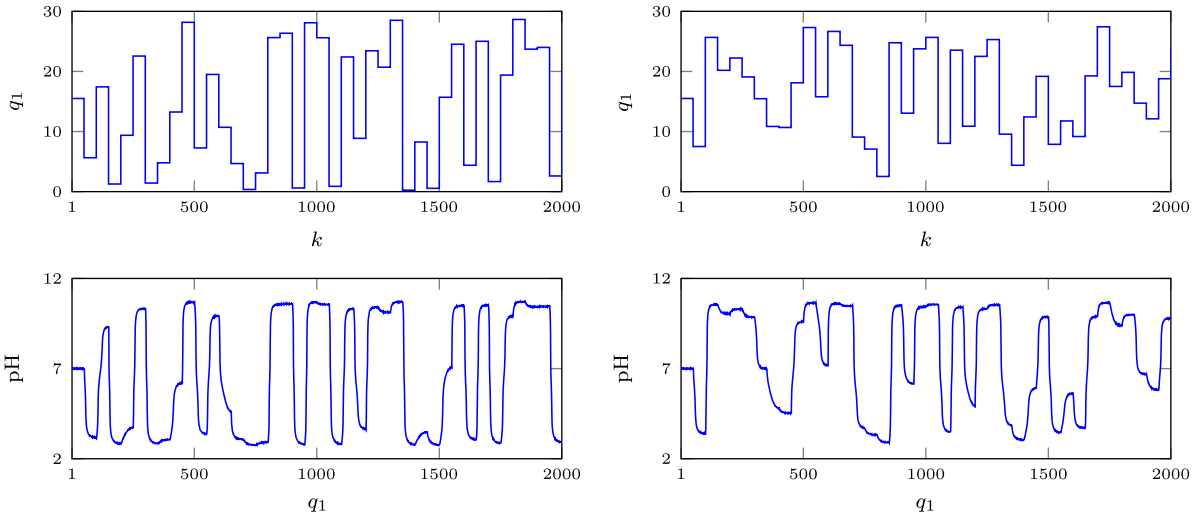

Fig. 5. The dynamic training data set (*left panels*) and the validation data set (*right panels*).

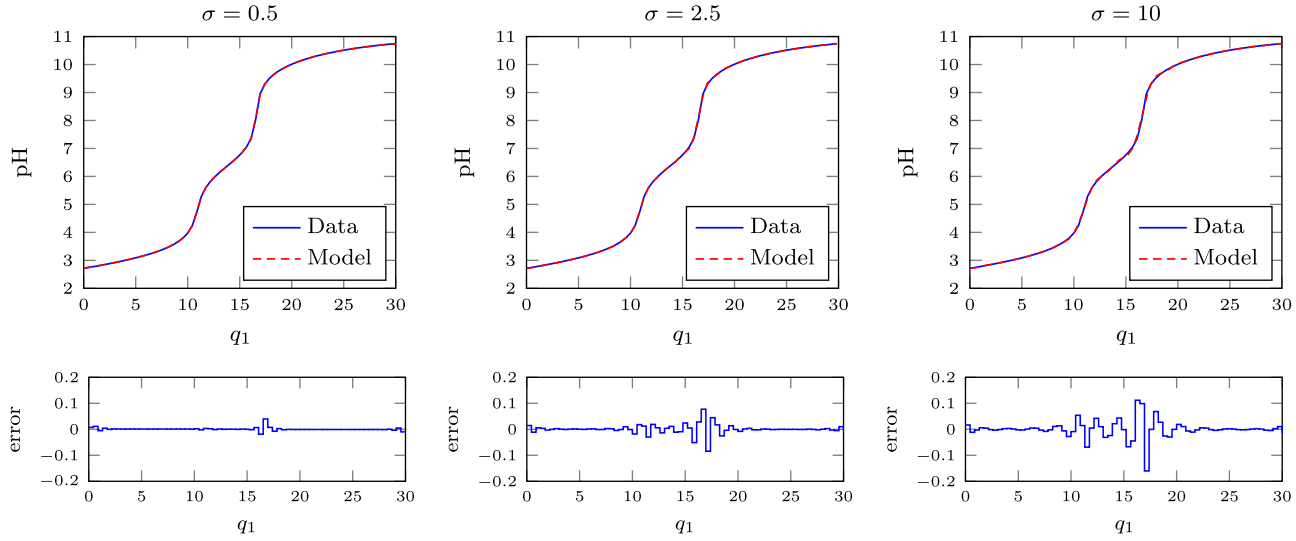

Fig. 6. The steady-state validation data set vs. the output of the LS-SVM approximator and model errors for $\sigma=0.5$ (*left panels*), $\sigma=2.5$ (*middle panels*), $\sigma=10$ (*right panels*); $n_{sv}=100$.

2.5679×10^{-3}, 2.3819×10^{-2} and 7.7876×10^{-2} for $\sigma=0.5$, $\sigma=2.5$ and $\sigma=10$, respectively. As shown in Fig. 7, the support vectors are located in a equidistant way in the domain of q_1, which is enforced by the steady-state training data set.

In step 2 of all three pruning algorithms of the LS-SVM approximator only one training pattern is removed from the training data set. Such an approach makes it possible to precisely compare three pruning algorithms. They are compared in terms of the error of the steady-state part of the Wiener model SSE_s (Eqs. (31)) and the error of the whole Wiener model SSE_d (Eqs. (32)). Fig. 8 compares the influence of the number of support vectors on the error of the steady-state part of the Wiener model SSE_s in three considered pruning algorithms. In order to demonstrate generalisation abilities of the models, the errors for the

validation data set are given. Since in the third pruning algorithm pruning is carried out taking into account the error of the complete dynamic Wiener model, its linear dynamic part must be found before pruning. The second-order of dynamics is used as in [25], hence the dynamic part (4) becomes

$$v(k)=b_1u(k-1)+b_2u(k-2)-a_1v(k-1)-a_2v(k-2)$$

For identification the prediction error identification method is used. For the full LS-SVM steady-state nonlinear part, the parameters of the dynamic linear one are $a_1=-6.6833\times10^{-1}$, $a_2=-2.0995\times10^{-1}$, $b_1=3.4400\times10^{-2}$, $b_2=9.1237\times10^{-2}$. Taking into account Fig. 8, it may be observed that in general the parameter σ influences the model error and its pruning possibility. The lower the value of σ, the more precise the LS-SVM approximation, but it needs more support vectors. It is also important to notice that in general for the same number of support vectors the second pruning algorithm gives the best result, much better than the first one, the third algorithm is somehow worse than the second one. In different words, the LS-SVM approximator with a fixed number of support vectors is most precise when pruned using the second method. For example, the LS-SVM Wiener model with $\sigma=5$ and 30 support vectors is characterised by the SSE_s error 1.0422×10^{0}, 8.9491×10^{-2} and 6.1411×10^{-1} when pruned by the algorithms 1, 2 and 3, respectively. For 20 support vectors the SSE_s error in the first algorithm increases to 2.6694×10^{2}, whereas the algorithms 2 and 3 give very moderate values 5.4809×10^{-1} and 6.5663×10^{-1}, respectively. Figs. 9 and 10 show the steady-state validation data set vs. the output of the pruned LS-SVM approximators and location of support vectors for $\sigma=0.5$ in two cases: when $n_{sv}=30$ and $n_{sv}=50$. For such a small value of σ 30 support vectors are insufficient, but the models with

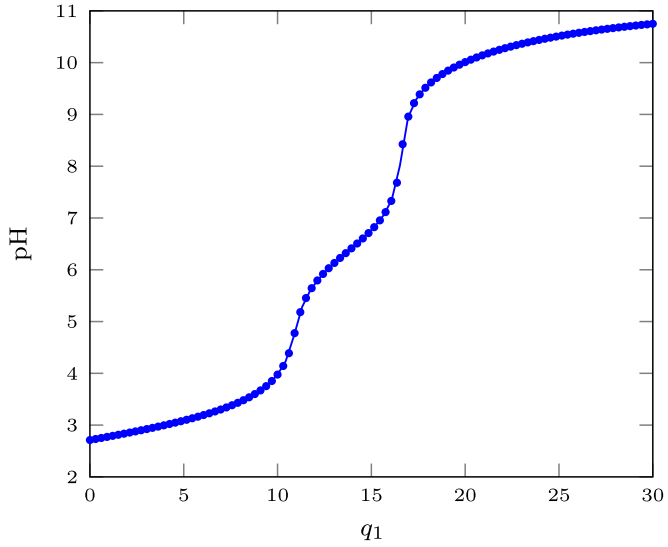

Fig. 7. The location of the support vectors of the full model ($n_{sv}=100$).

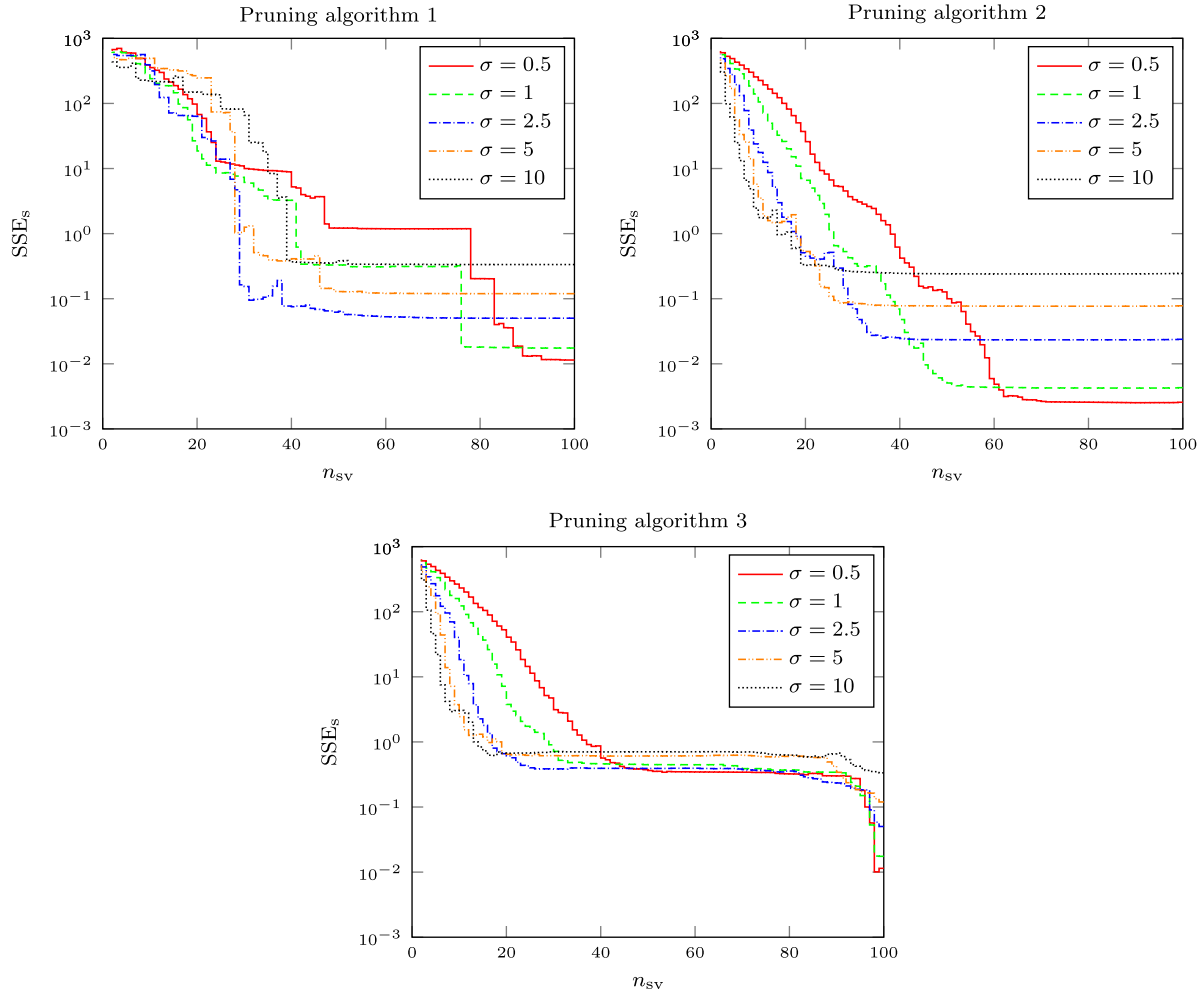

Fig. 8. The influence of the number of support vectors n_{sv} on the error SSE_s of the steady-state part of the LS-SVM Wiener model for the validation data set in three compared pruning algorithms.

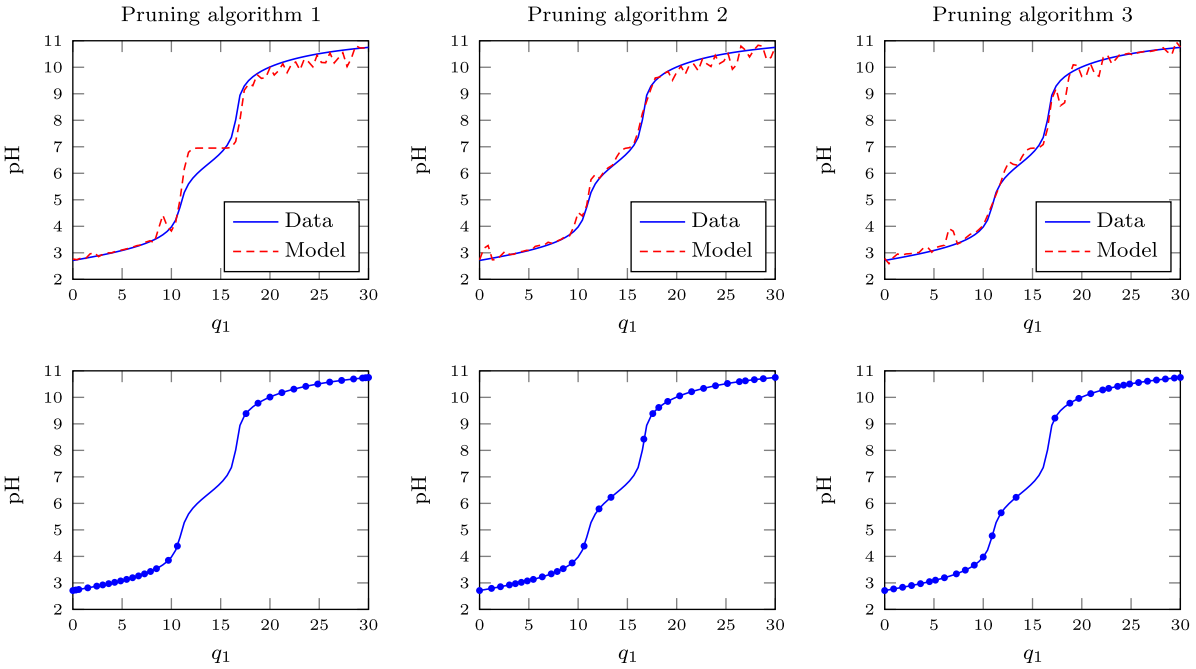

Fig. 9. The steady-state validation data set vs. the output of the pruned LS-SVM approximators (*top panels*) and location of support vectors (*bottom panels*) obtained by three pruning algorithms; $\sigma = 0.5$, $n_{sv} = 30$.

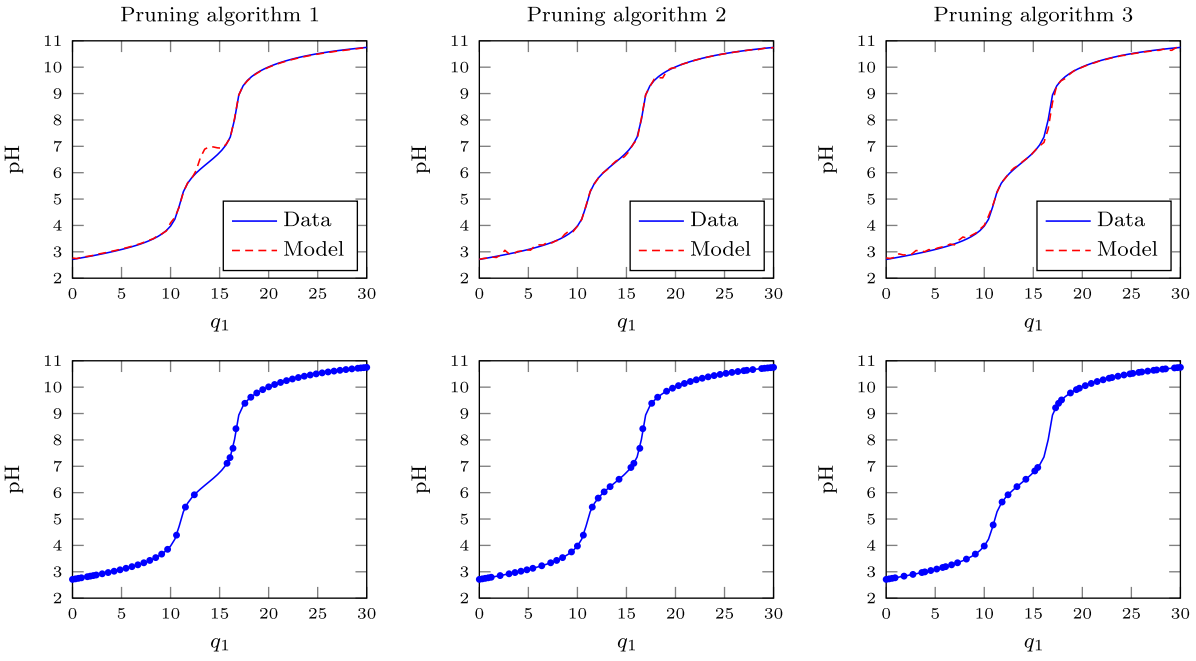

Fig. 10. The steady-state validation data set vs. the output of the pruned LS-SVM approximators (*top panels*) and location of support vectors (*bottom panels*) obtained by three pruning algorithms; $\sigma = 0.5$, $n_{sv} = 50$.

50 vectors are also not perfect. The first pruning method unnecessarily reduces the support vectors in the central part of the steady-state characteristics.When $\sigma = 5$ the second and the third pruning algorithms give quite good models with only 20 support vectors as depicted in Fig. 11, the first algorithm gives big errors. For $\sigma = 5$ the models with 30 support vectors obtained by the pruning algorithms 2 and 3 are very good as shown in Fig. 12 (the second algorithm in fact gives better approximation), the algorithm 1 still needs more support vectors. Too big values of the parameter σ are rather disadvantageous, because the model accuracy deteriorates as shown in Fig. 13 for the LS-SVM approximator with $\sigma = 10$ and 30 support vectors.

Next, the complete Wiener models with the pruned LS-SVM steady-state approximators are assessed using the dynamic validation data set shown in Fig. 5. Fig. 14 compares the influence of the number of support vectors on the error SSE_d of the LS-SVM Wiener model for the validation data set in three considered pruning algorithms. Relation between the number of support vectors and model error SSE_d and effectiveness of three pruning algorithms is quite similar to the case when the error SSE_s of the steady-state part is only considered (Fig. 8). In general, the second

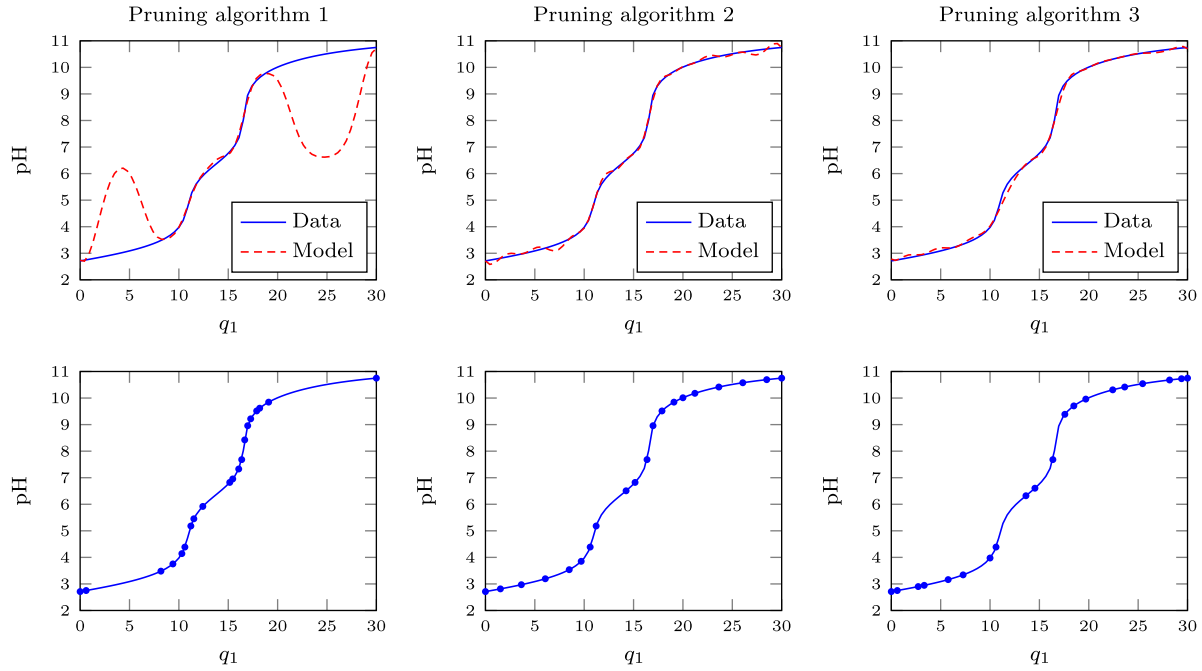

Fig. 11. The steady-state validation data set vs. the output of the pruned LS-SVM approximators (*top panels*) and location of support vectors (*bottom panels*) obtained by three pruning algorithms; $\sigma = 5$, $n_{sv} = 20$.

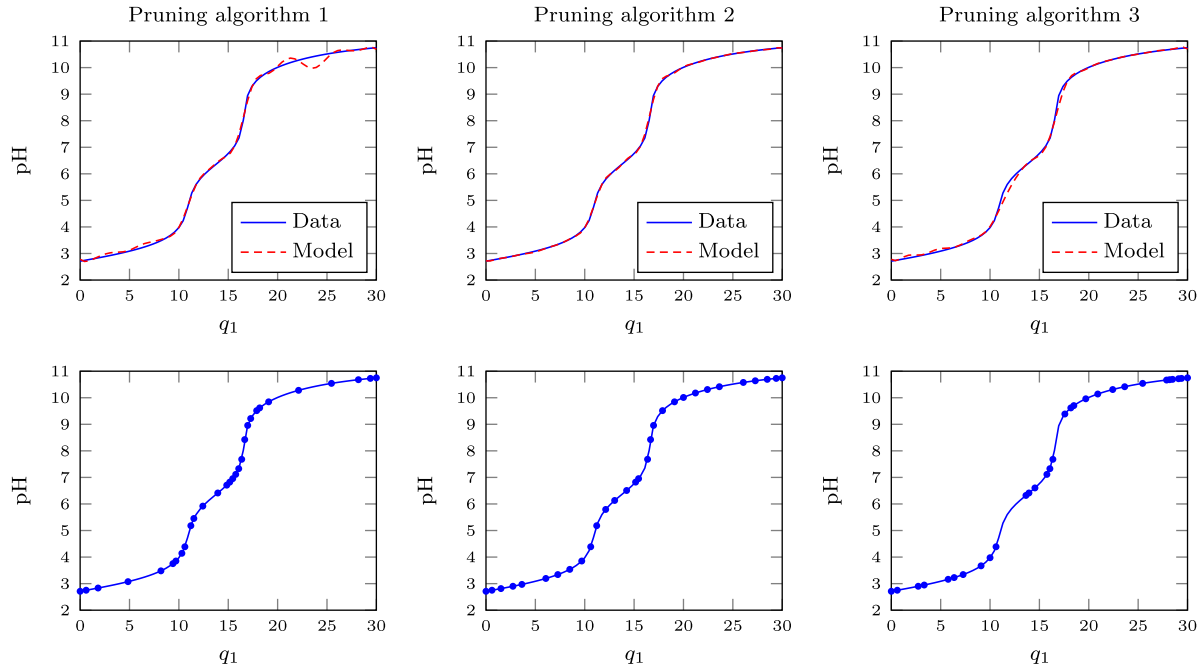

Fig. 12. The steady-state validation data set vs. the output of the pruned LS-SVM approximators (*top panels*) and location of support vectors (*bottom panels*) obtained by three pruning algorithms; $\sigma = 5$, $n_{sv} = 30$.

pruning algorithm gives the best results, the third one is a little bit worse whereas the first one generates very imprecise approximators. For example, the LS-SVM Wiener model with $\sigma = 5$ and 30 support vectors is characterised by the SSE_d error 9.0679×10^1, 6.7043×10^1 and 7.5833×10^1 when pruned by the algorithms 1, 2 and 3, respectively. For 20 support vectors the SSE_d error in the first algorithm increases to 7.5584×10^3, whereas the algorithms 2 and 3 give very moderate values 7.6340×10^1 and 7.6824×10^1, respectively. The role of the parameter σ is the same when only the steady-state part of the model is analysed.

As a good compromise between accuracy of the steady-state part, its smoothness, accuracy of the full LS-SVM Wiener model and model complexity determined by the number of support vectors, the models with $\sigma = 5$ are chosen. Finally, it is interesting to compare the dynamic validation data set and the output of the pruned models when pruned by means of pruning algorithms 1 and 2 (the algorithm 3 gives comparable results to algorithm 2). When the number of support vectors is 20 the first pruning algorithm gives a model which is unable to follow the output signal of the process whereas the model obtained by the second pruning algorithm performs quite well as depicted in Fig. 15. For

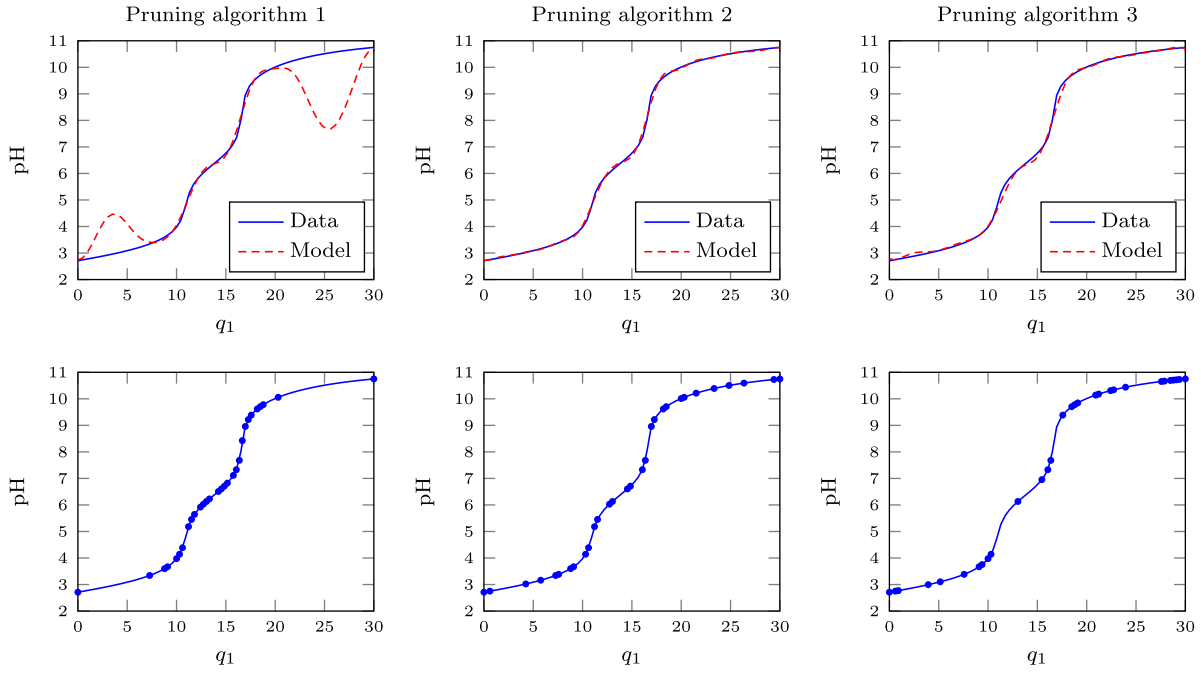

Fig. 13. The steady-state validation data set vs. the output of the pruned LS-SVM approximators (*top panels*) and location of support vectors (*bottom panels*) obtained by three pruning algorithms; $\sigma = 10$, $n_{sv} = 30$.

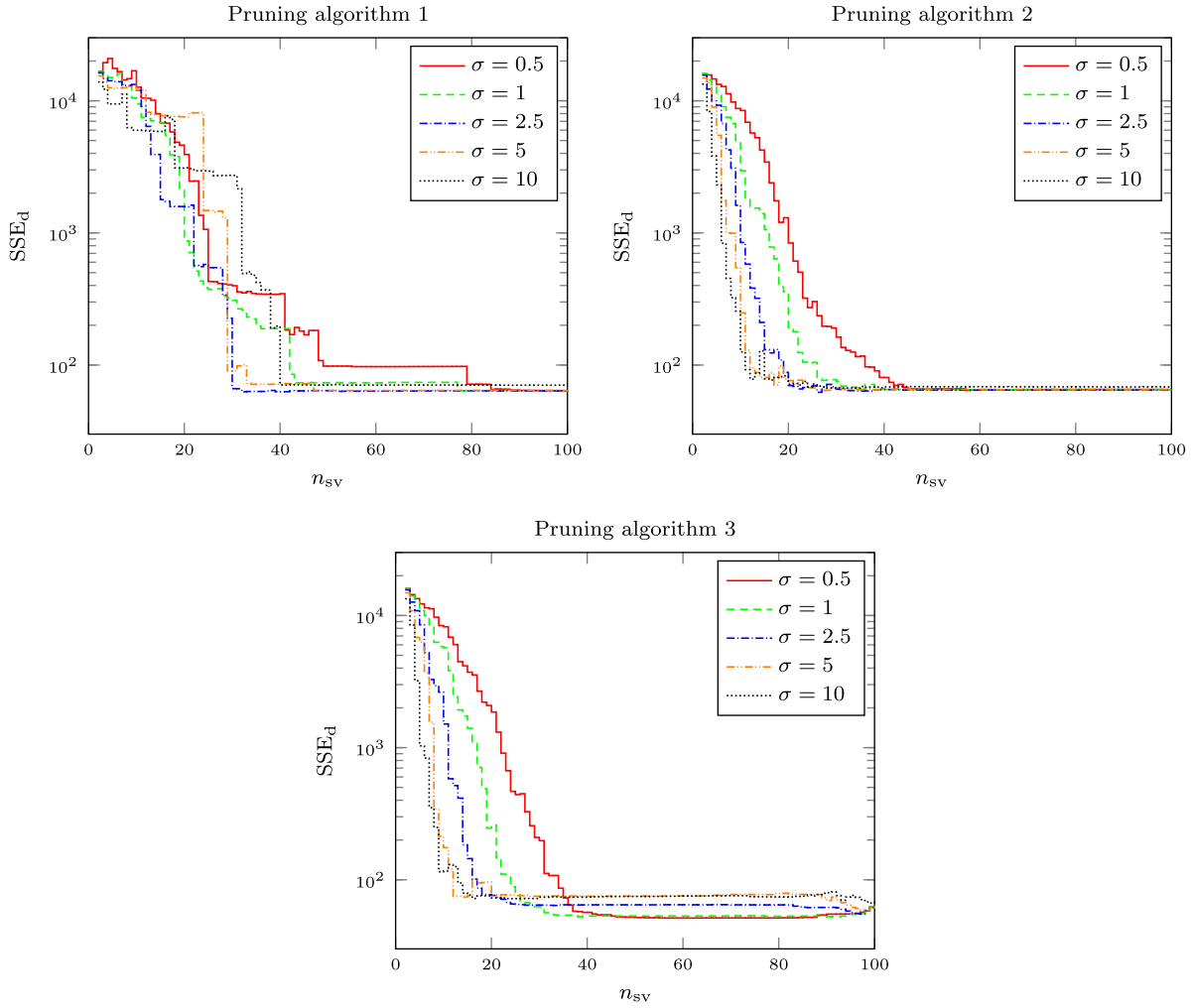

Fig. 14. The influence of the number of support vectors n_{sv} on the error $\mathrm{SSE_d}$ of the LS-SVM Wiener model for the validation data set in three compared pruning algorithms.

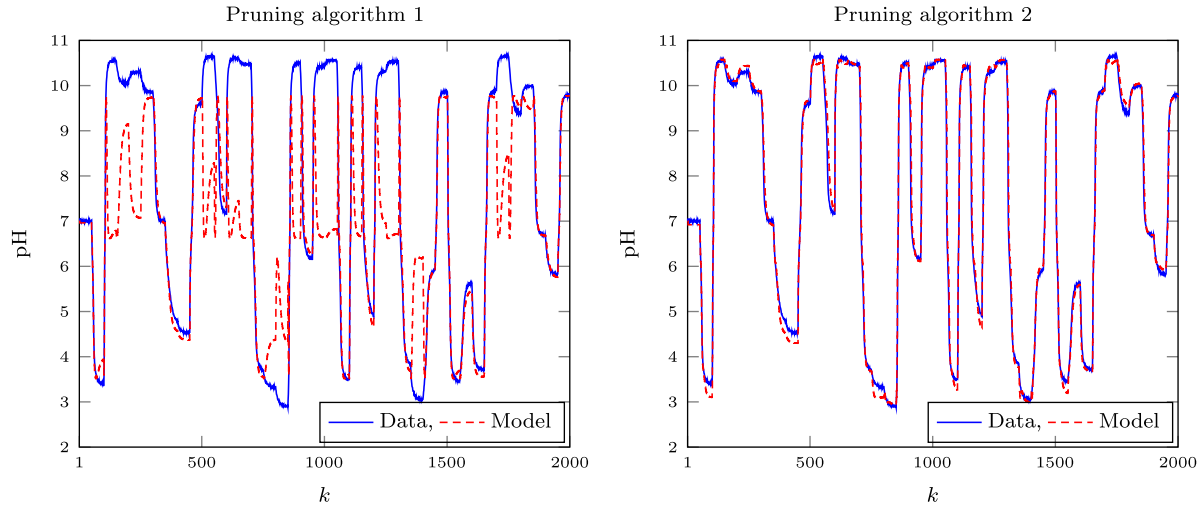

Fig. 15. The dynamic validation data set (*solid line*) vs. the output of the pruned models obtained by pruning algorithms 1 and 2 (*dashed line*); $\sigma = 5$, $n_{sv} = 20$.

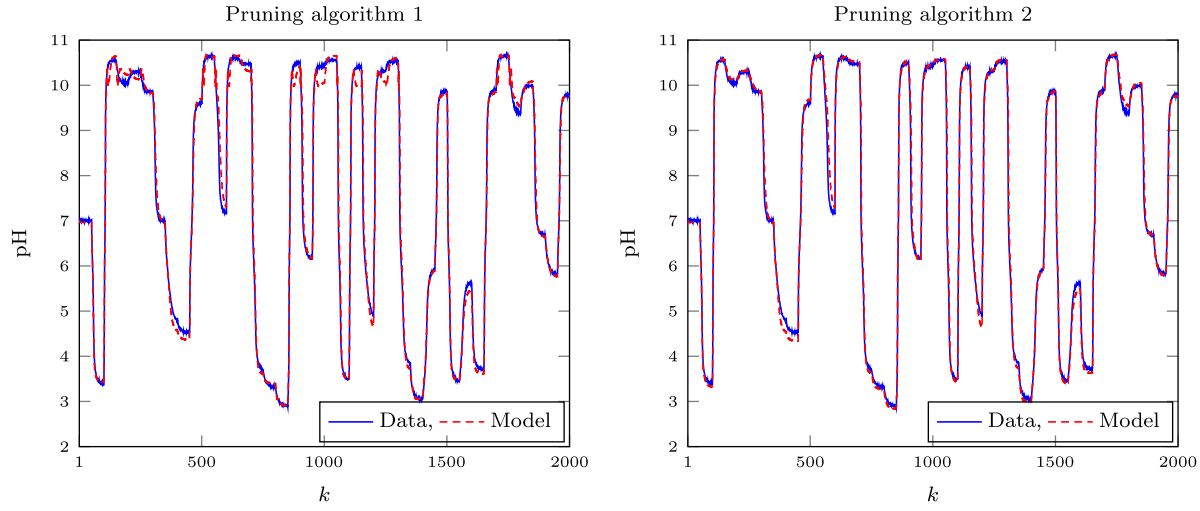

Fig. 16. The dynamic validation data set (*solid line*) vs. the output of the pruned models obtained by pruning algorithms 1 and 2 (*dashed line*); $\sigma = 5$, $n_{sv} = 30$.

30 support vectors the model obtained by the first pruning method gives acceptable prediction, but the model pruned by the second method is almost as good as in case of the full (not pruned) model with 100 support vectors as shown in Figs. 16 and 17, respectively. It is an interesting question whether or not it is desirable to recalculate the parameters of the linear dynamic part of the Wiener model after pruning. Taking into account the most efficient second pruning algorithm, for $\sigma = 2.5$, $\sigma = 5$ and $\sigma = 10$ differences between the model with the recalculated linear part and a constant one in terms of the SSE_d error are very small (below 0.1%). Some differences are observed for small values of the parameter σ. For example, for 30 support vectors such a difference in the SSE_d error is 3.4% and 0.7% for $\sigma = 1$ and $\sigma = 0.5$, respectively.

6.3. *Predictive control of neutralisation process*

In this section predictive control of the neutralisation reactor is discussed. At first it is interesting to compare efficiency of the "ideal" MPC-NO algorithm with nonlinear optimisation repeated at each sampling instant. Three models of the process are used for prediction: the full (not pruned) LS-SVM Wiener model with 100 support vectors as well as the pruned models with 20 and 30 support vectors. Pruning is carried out by the most efficient second method, the models trained for $\sigma = 5$ are used. Fig. 18 compares

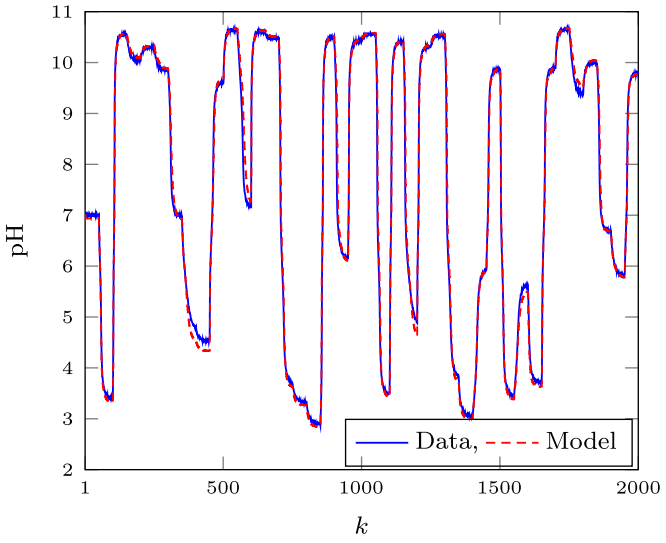

Fig. 17. The dynamic validation data set (*solid line*) vs. the output of the full model, $n_{sv} = 100$ (*dashed line*); $\sigma = 5$.

the trajectories obtained when the full model and the model with 20 support vectors are used. Because the pruned model performs quite well for steady-state and dynamic data (Figs. 11 and 15,

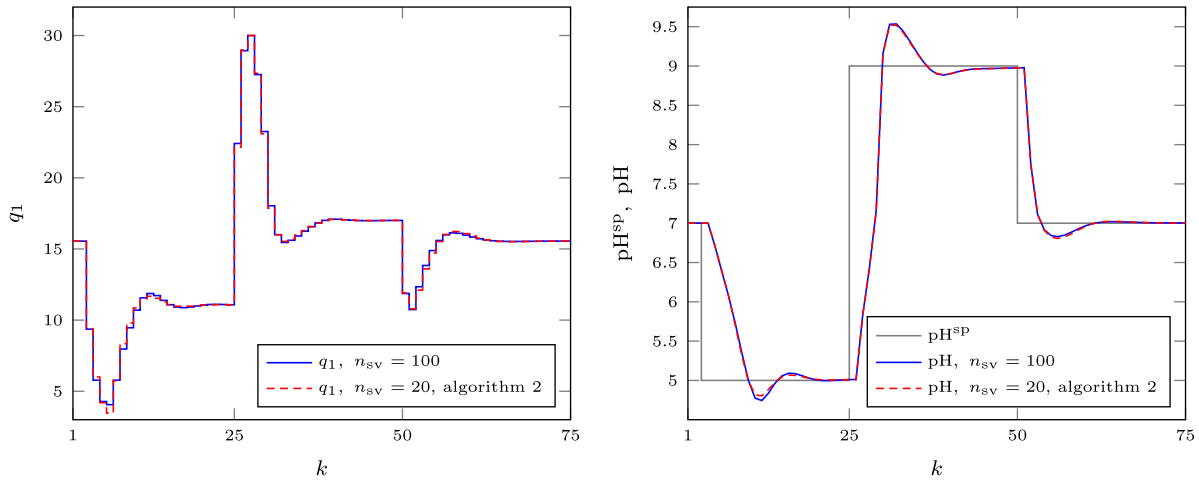

Fig. 18. Simulation results of the MPC-NO algorithm based on: the full LS-SVM Wiener model with $n_{sv}=100$ support vectors (*solid line*) and the model with $n_{sv}=20$ vectors pruned by the algorithm 2 (*dashed line*); $\sigma=5$.

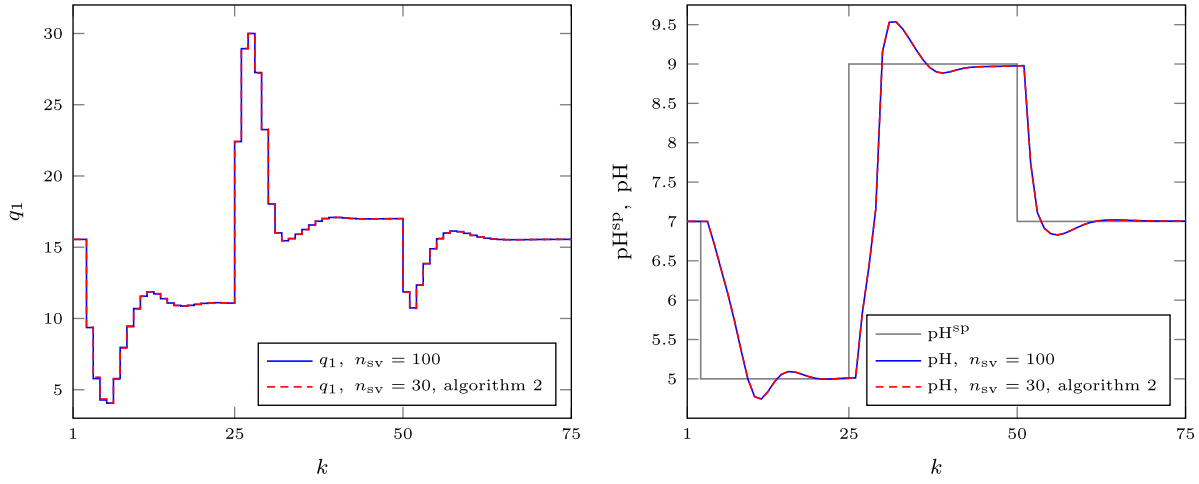

Fig. 19. Simulation results of the MPC-NO algorithm based on: the full LS-SVM Wiener model with $n_{sv}=100$ support vectors (*solid line*) and the model with $n_{sv}=30$ vectors pruned by the algorithm 2 (*dashed line*); $\sigma=5$.

respectively), the differences between the "ideal" trajectories (corresponding to the full model) are very small. The LS-SVM Wiener model with 30 support vectors has excellent approximation abilities (Figs. 12 and 16) and when it is used in MPC it makes it possible to obtain practically the same trajectories as in the case of the full model which is illustrated in Fig. 19. That is why for further application in MPC the LS-SVM model with 30 support vectors obtained for $\sigma=5$ and pruned by means of the second algorithm is chosen.

Next, two MPC algorithms are compared: the discussed MPC-NPLPT algorithm with on-line trajectory linearisation and quadratic optimisation and the MPC-NO algorithm with on-line nonlinear optimisation. For optimisation the active set method is used in the first case and the Sequential Quadratic Programming (SQP) algorithm is used in the second case [29]. Both MPC algorithms use on-line the same nonlinear LS-SVM Wiener model with 30 support vectors pruned by the second algorithms. Tuning parameters of all algorithms are the same and the same set-point trajectories are considered. All simulations are carried out in Matlab. The common parameters of both algorithms are the same: $N=10$, $N_u=3$, $\lambda=0.05$, the constraints imposed on the manipulated variable are $q_1^{min}=0$ ml/s, $q_1^{max}=30$ ml/s, the additional parameters of the MPC-NPLPT algorithm are $t_{max}=5$, $N_0=3$ and in order to demonstrate the role of internal iterations as many as 5 different values of the parameters $\delta_u=\delta_y$ are assumed: 10, 1, 0.1, 0.01, 0.001. The same set-point trajectory as in [25] is used. The obtained simulation results are depicted in Figs. 20, 21 and 22 for the tuning parameters $\delta_u=\delta_y$ equal to 10, 1 and 0.1, respectively. The MPC-NPLPT algorithm with $\delta_u=\delta_y=10$ gives quite significantly different trajectories than the "ideal" MPC-NO one because the overshoot is bigger. Reducing the tuning parameters to 1 leads to decreasing the discrepancy between the trajectories of the compared algorithms. Finally, for $\delta_u=\delta_y=0.1$ the MPC-NPLPT algorithm gives practically the same trajectories as the MPC-NO one. Although taking into account accuracy of the MPC-NPLPT algorithm it is desirable to use small values of the additional tuning parameters δ_u and δ_y, the smaller their values, the more internal iterations are necessary. For $\delta_u=\delta_y=10$ the algorithm needs 80 iterations (for the whole simulation horizon), for $\delta_u=\delta_y=1$ it needs 96 iterations and for $\delta_u=\delta_y=0.1$ as many as 114 ones. The number of internal iterations in the consecutive sampling instants of the MPC-NPLPT algorithm for different values of $\delta_u=\delta_y$ is shown in Fig. 23 (one internal iteration is in fact the main task of the algorithm).

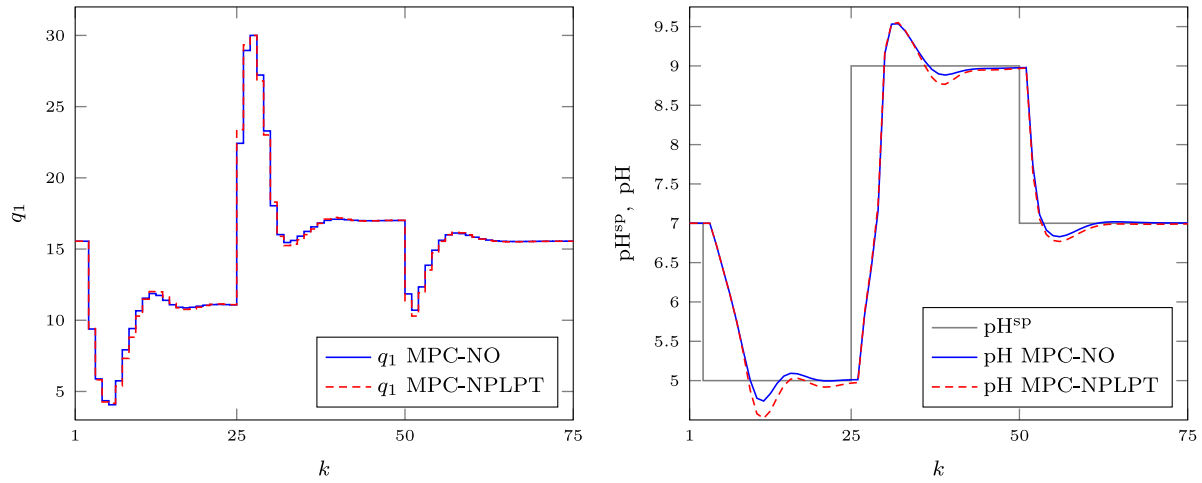

Fig. 20. Simulation results: the MPC-NO algorithm (*solid line*) vs. the MPC-NPLPT algorithm with $\delta_u = \delta_y = 10$ (*dashed line*); both algorithms use the same LS-SVM Wiener model with $\sigma = 5$, $n_{sv} = 30$, pruned by the algorithm 2.

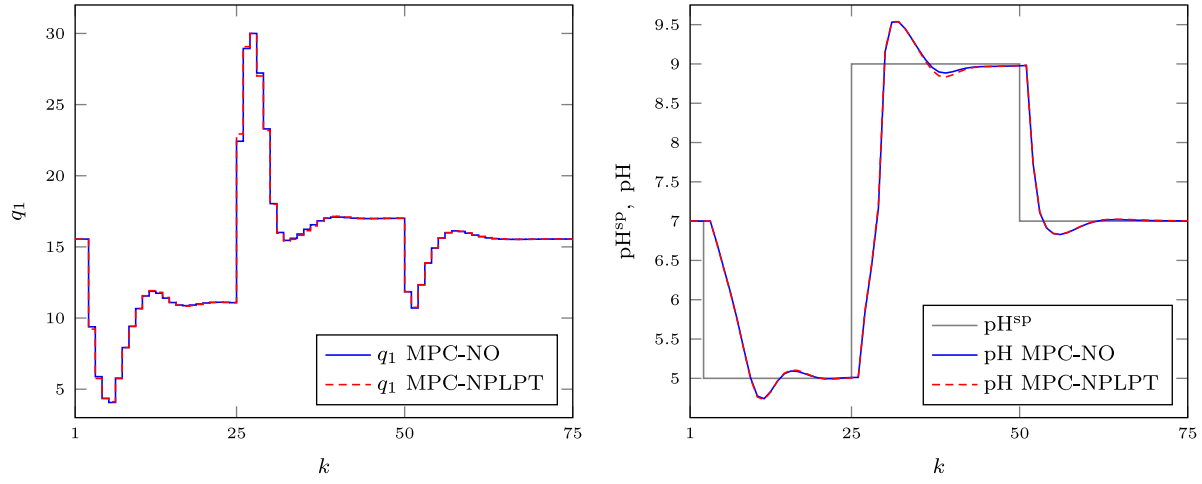

Fig. 21. Simulation results: the MPC-NO algorithm (*solid line*) vs. the MPC-NPLPT algorithm with $\delta_u = \delta_y = 1$ (*dashed line*); both algorithms use the same LS-SVM Wiener model with $\sigma = 5$, $n_{sv} = 30$, pruned by the algorithm 2.

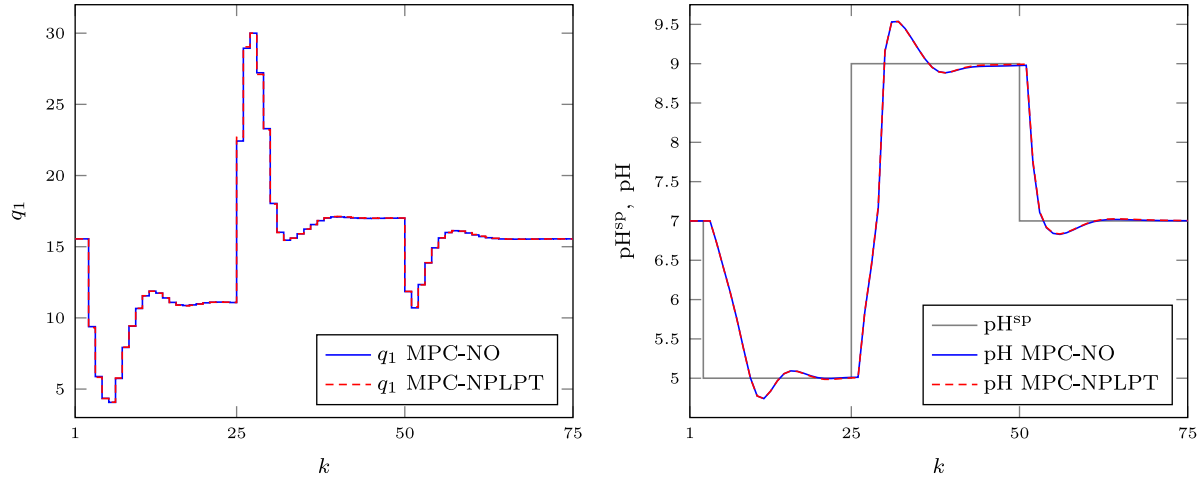

Fig. 22. Simulation results: the MPC-NO algorithm (*solid line*) vs. the MPC-NPLPT algorithm with $\delta_u = \delta_y = 0.1$ (*dashed line*); both algorithms use the same LS-SVM Wiener model with $\sigma = 5$, $n_{sv} = 30$, pruned by the algorithm 2.

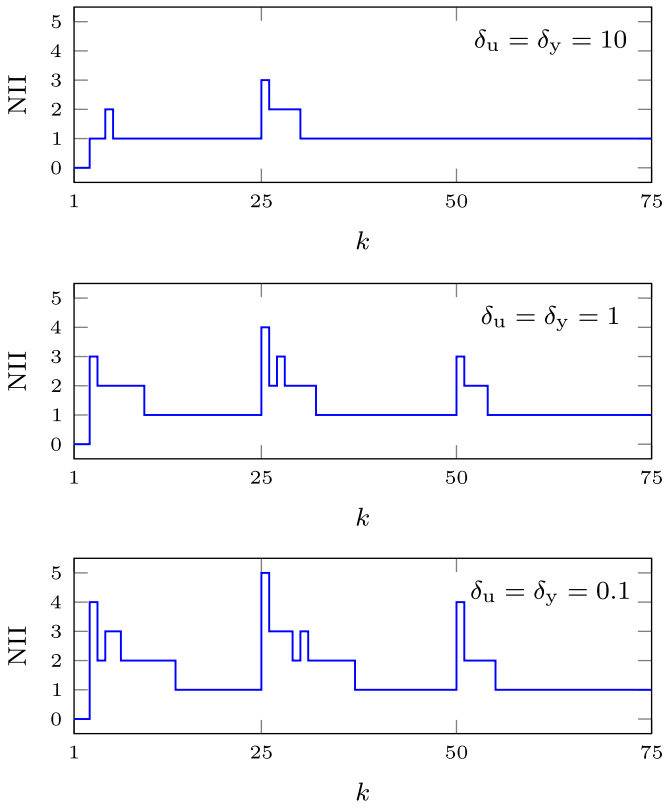

Fig. 23. The number of internal iterations NII in the consecutive sampling instants of the MPC-NPLPT algorithm for different values of $\delta_u = \delta_y$; all algorithms use the same LS-SVM Wiener model with $\sigma = 5$, $n_{sv} = 30$, pruned by the algorithm 2.

Table 1
Accuracy criterion $D_1 = D_2$ obtained in the MPC-NPLPT and MPC-NO algorithms; all algorithms use the same full LS-SVM Wiener model with $\sigma = 5$, $n_{sv} = 100$.

Algorithm	$\delta_u = \delta_u$	Internal iterations	$D_1 = D_2$
MPC-NPLPT	10	80	3.5784×10^{-1}
MPC-NPLPT	1	95	1.4752×10^{-2}
MPC-NPLPT	0.1	114	4.7970×10^{-3}
MPC-NPLPT	0.01	136	1.5414×10^{-3}
MPC-NPLPT	0.001	148	1.3430×10^{-3}
MPC-NO	–	–	0

In order to further analyse accuracy of the MPC-NPLPT algorithm the following accuracy criterion is defined

$$D_1 = \sum_{k=1}^{75} (\mathrm{pH}_{\mathrm{MPC-NO}}(k) - \mathrm{pH}_{\mathrm{MPC-NPLPT}}(k))^2 \tag{40}$$

which describes the discrepancy between the trajectories obtained in the MPC-NO algorithm ($\mathrm{pH}_{\mathrm{MPC-NO}}$) and the MPC-NPLPT one ($\mathrm{pH}_{\mathrm{MPC-NPLPT}}$). Both algorithms use the same model (full or pruned). The second criterion

$$D_2 = \sum_{k=1}^{75} (\mathrm{pH}^{n_{sv}=100}_{\mathrm{MPC-NO}}(k) - \mathrm{pH}^{n_{sv}\le 100}_{\mathrm{MPC-NPLPT}}(k))^2 \tag{41}$$

measures the discrepancy between the trajectories obtained in the MPC-NO algorithm based on the full model ($\mathrm{pH}^{n_{sv}=100}_{\mathrm{MPC-NO}}$) and the MPC-NPLPT algorithm based on a pruned model ($\mathrm{pH}^{n_{sv}\le 100}_{\mathrm{MPC-NPLPT}}$). When $n_{sv} = 100$, $D_1 = D_2$. At first, in Table 1, the values of D_1 are given for MPC-NO and MPC-NPLPT algorithms based on the same full LS-SVM Wiener model with $\sigma = 5$ and 100 support vectors. The smaller the values of δ_u and δ_y, the smaller the discrepancy, but more internal iterations are necessary. Table 2 gives the values of D_1 and D_2 when the LS-SVM Wiener model pruned by algorithm 2 with only 30 support vectors is used. One can easily notice that for the consecutive values of δ_u and δ_y the MPC-NPLPT algorithm makes it possible to achieve very good control accuracy when it uses the pruned model, comparable to that obtained when the full model is used.

Table 2
Accuracy criteria D_1 and D_2 obtained in the MPC-NPLPT and MPC-NO algorithms; all algorithms use the same LS-SVM Wiener model with $\sigma = 5$, $n_{sv} = 30$, pruned by the algorithm 2.

Algorithm	$\delta_u = \delta_u$	Internal iterations	D_1	D_2
MPC-NPLPT	10	80	3.5331×10^{-1}	3.5687×10^{-1}
MPC-NPLPT	1	96	1.4289×10^{-2}	1.4345×10^{-2}
MPC-NPLPT	0.1	114	4.8176×10^{-3}	4.7275×10^{-3}
MPC-NPLPT	0.01	136	1.2389×10^{-3}	1.5024×10^{-3}
MPC-NPLPT	0.001	151	1.1176×10^{-3}	1.4259×10^{-3}
MPC-NO	–	–	0	3.8537×10^{-4}

Finally, computational complexity of the MPC-NPLPT and MPC-NO algorithms is compared (it is assessed using the cputime command in Matlab). Table 3 gives scaled computational burden for the MPC algorithms which use the same full LS-SVM Wiener model with 100 support vectors, whereas Table 4 shows scaled computational burden for MPC algorithms in which the same model with only 30 support vectors are used, pruned by the algorithm 2. In both cases $\sigma = 5$. The following lengths of the control horizon are considered: 1, 2, 3, 4, 5 and 10 as it has a predominant influence on computational complexity, $N = 10$. All results are scaled in such a way that computational burden for the MPC-NO algorithm based on the full LS-SVM Wiener model and with $N_u = 10$ is 100%. Taking into account the obtained numerical values, two conclusions may be drawn. Firstly, the MPC-NPLPT algorithm with quadratic optimisation is many times less computationally demanding than the MPC-NO one. For example, for $N_u = 3$ and $\delta_u = \delta_y = 0.1$ some 6 times, for longer control horizons that comparison is even better for the MPC-NPLPT algorithm. Secondly, for the same set of parameters the MPC-NPLPT algorithm based on the pruned model is some 3 times less demanding than the same algorithm based on the full model. Finally, it may be observed that the mentioned factors of computational complexity of the MPC-NPLPT algorithm and the use of a reduced model are approximately the same for different control horizons and values of δ_u, δ_y, the only difference is when the longest control horizon is considered.

7. Conclusions

This paper describes a computationally efficient nonlinear MPC algorithm for the neutralisation process based on a pruned LS-SVM Wiener model.

Firstly, since the classical LS-SVM approximator is usually characterised by an excessive number of parameters, the model containing as many as 100 support vectors (105 parameters) is pruned using three different methods. In particular, the introduced pruning methods, in which the support vectors are removed taking into account directly the error of the model, not the error minimised during training the LS-SVM approximator, turn out to give very good models in terms of accuracy and sparseness. It is possible to remove as much as 70% of support vectors without any significant deterioration of model accuracy.

Secondly, the LS-SVM Wiener model is used in an MPC algorithm with on-line successive linearisation of the predicted trajectory. Such an approach makes it possible to find at each

Table 3
Scaled computational burden of the MPC-NPLPT and MPC-NO algorithms; all algorithms use the same full LS-SVM Wiener model with $\sigma = 5$, $n_{sv} = 100$.

Algorithm	$\delta_u = \delta_y$	$N_u = 1$	$N_u = 2$	$N_u = 3$	$N_u = 4$	$N_u = 5$	$N_u = 10$
MPC-NPLPT	10	0.67%	1.09%	1.46%	1.84%	2.17%	3.42%
MPC-NPLPT	1	0.84%	1.29%	1.74%	2.13%	2.57%	4.04%
MPC-NPLPT	0.1	0.99%	1.46%	2.08%	2.56%	3.05%	4.75%
MPC-NPLPT	0.01	1.19%	1.86%	2.55%	3.19%	3.74%	5.87%
MPC-NPLPT	0.001	1.39%	2.16%	2.91%	3.62%	4.38%	6.74%
MPC-NO	–	3.44%	6.87%	12.23%	18.05%	27.72%	100.00%

Table 4
Scaled computational burden of the MPC-NPLPT and MPC-NO algorithms; all algorithms use the same LS-SVM Wiener model with $\sigma = 5$, $n_{sv} = 30$, pruned by the algorithm 2.

Algorithm	$\delta_u = \delta_y$	$N_u = 1$	$N_u = 2$	$N_u = 3$	$N_u = 4$	$N_u = 5$	$N_u = 10$
MPC-NPLPT	10	0.22%	0.36%	0.49%	0.64%	0.78%	1.53%
MPC-NPLPT	1	0.27%	0.42%	0.59%	0.75%	0.92%	1.81%
MPC-NPLPT	0.1	0.33%	0.48%	0.70%	0.88%	1.09%	2.09%
MPC-NPLPT	0.01	0.39%	0.62%	0.86%	1.11%	1.35%	2.61%
MPC-NPLPT	0.001	0.45%	0.72%	0.98%	1.25%	1.54%	2.98%
MPC-NO	–	1.18%	2.52%	4.29%	6.61%	10.37%	40.92%

sampling instant the future control scenario from a series of easy to solve quadratic optimisation problems. Unlike many MPC schemes for Wiener systems, e.g. [6,13,32], the inverse model of the steady-state part is not used. It is shown that the trajectories of the discussed MPC algorithm with on-line linearisation are practically the same as those possible in the MPC strategy with nonlinear optimisation repeated at each sampling instant on-line. For the chosen tuning parameters of MPC and the set-point scenario, the discussed MPC algorithm is approximately 6 times less computationally demanding than the "ideal" MPC with nonlinear optimisation. Furthermore, model pruning does not lead to any significant negative effect on quality of control, i.e. MPC algorithms based on the pruned model give almost the same performance as the algorithms in which the full LS-SVM Wiener model with as many as 100 support vectors is used. Model pruning makes it possible to further reduce computational effort some 3 times.

Finally, it is necessary to emphasise the fact that the described model pruning technique and the MPC algorithm may be applied to many other dynamic systems represented by the LS-SVM Wiener model. The cascade Wiener structure is very universal, it may be successfully used to describe numerous processes, e.g. chemical reactors, distillation columns, separation processes, gasifiers, hydraulic systems and even biomedical systems (the relaxation process during anaesthesia), an excellent review is given in [18]. Furthermore, the LS-SVM approximator may be efficiently used as a steady-state nonlinear part of the Wiener model since it has excellent approximation abilities and it may be found easily. It is important to notice that application of the LS-SVM approximator makes on-line trajectory linearisation possible, which may be not true for other model structures. Extension of the presented pruning and MPC algorithms for multiple-input multiple-output processes is straightforward. The described computationally efficient MPC algorithm based on the optimised LS-SVM model may be used in the multilayer control system structure and in networked industrial process control [42,44]. Moreover, it is possible to use the described pruning algorithm to optimise the number of parameters in alternative model structures, e.g. in a fuzzy dynamic model [35] and also in a fuzzy dynamic model of a distributed parameter system [34]. The mentioned issues may be considered during the future works.

Acknowledgement

The work presented in this paper was supported by the Polish national budget funds for science.

References

[1] A.S. Al-Araji, M.F. Abbod, H.S. Al-Raweshidy, Applying posture identifier in designing an adaptive nonlinear predictive controller for nonholonomic mobile robot, Neurocomputing 99 (2013) 543–554.
[2] N.L. Azad, A. Mozaffari, J.K. Hedrick, A hybrid switching predictive controller based on bi-level kernel-based ELM and online trajectory builder for automotive coldstart emissions reduction, Neurocomputing 173 (2016) 1124–1141.
[3] B.M. Åkesson, H.T. Toivonen, J.B. Waller, R.H. Nyström, Neural network approximation of a nonlinear model predictive controller applied to a pH neutralization process, Comput. Chem. Eng. 29 (2005) 323–335.
[4] N. Bigdeli, M. Haeri, Predictive functional control for active queue management in congested TCP/IP networks, ISA Trans. 48 (2009) 107–121.
[5] J.M. Böling, D.E. Seborg, J.P. Hespanha, Multi-model adaptive control of a simulated pH neutralization process, Control Eng. Pract. 15 (2007) 663–672.
[6] A.L. Cervantes, O.E. Agamennoni, J.L. Figueroa, A nonlinear model predictive control based on Wiener piecewise linear models, J. Process Control 13 (2003) 655–666.
[7] J. Chen, Y. Peng, W. Han, M. Guo, Adaptive fuzzy sliding mode control in PH neutralization process, Procedia Eng. 15 (2011) 954–958.
[8] D. Dougherty, D. Cooper, A practical multiple model adaptive strategy for single-loop MPC, Control Eng. Pract. 11 (2003) 141–159.
[9] R. Dubay, M. Hassan, C. Li, M. Charest, Finite element based model predictive control for active vibration suppression of a one-link flexible manipulator, ISA Trans. 53 (2014) 1609–1619.
[10] M.J. Fuente, C. Robles, O. Casado, S. Syafiie, F. Tadeo, Fuzzy control of a neutralization process, Eng. Appl. Artif. Intell. 19 (2006) 905–914.
[11] O. Galán, J.A. Romagnoli, A. Palazoglu, Real-time implementation of multilinear model-based control strategies-an application to a bench-scale pH neutralization reactor, J. Process Control 14 (2004) 571–579.
[12] F. Giri, E.W. Bai (Eds.), Block-oriented Nonlinear System Identification, Lecture Notes in Control and Information Sciences, vol. 404, Springer, Berlin, 2010.
[13] J.C. Gómez, A. Jutan, E. Baeyens, Wiener model identification and predictive control of a pH neutralisation process, Proc. IEE, Part D, Control Theory Appl. 151 (2004) 329–338.
[14] A. Grancharova, J. Kocijan, T.A. Johansen, Explicit output-feedback nonlinear predictive control based on black-box models, Eng. Appl. Artif. Intell. 24 (2011) 388–397.
[15] M. Henson, D. Seborg, Adaptive nonlinear control of a pH neutralization process, IEEE Trans. Control Syst. Technol. 2 (1994) 169–182.
[16] A.W. Hermansson, S. Syafiie, Model predictive control of pH neutralization processes: a review, Control Eng. Pract. 45 (2015) 98–109.
[17] S. Iplikci, A support vector machine based control application to the experimental three-tank system, Neurocomputing 49 (2010) 376–386.
[18] A. Janczak, Identification of Nonlinear Systems Using Neural Networks and Polynomial Models, A Block-Oriented Approach, Lecture Notes in Control and Information Sciences, vol. 310, Springer, Berlin, 2004.

[19] O. Karasakal, M. Guzelkaya, I. Eksin, E. Yesil, T. Kumbasar, Online tuning of fuzzy PID controllers via rule weighing based on normalized acceleration, Eng. Appl. Artif. Intell. 26 (2013) 184–197.
[20] B.J. de Kruif, T.J.A. de Vries, Pruning error minimization in least-squares support vector machines, IEEE Trans. Neural Netw. 14 (2003) 696–702.
[21] T. Kumbasar, I. Eksin, M. Guzelkaya, E. Yesil, Type-2 fuzzy model based controller design for neutralization processes, ISA Trans. 51 (2014) 277–287.
[22] N.R. Lakshmi Narayanan, P.R. Krishnaswamy, G.P. Rangaiah, An adaptive internal model control strategy for pH neutralization, Chem. Eng. Sci. 52 (1997) 3067–3074.
[23] A.P. Loh, K.O. Looi, K.F. Fong, Neural network modeling and control strategies for a pH process, J. Process Control 5 (1995) 355–362.
[24] M. Ławryńczuk, Computationally Efficient Model Predictive Control Algorithms: A Neural Network Approach, Studies in Systems, Decision and Control, vol. 3, Springer, Heidelberg, 2014.
[25] M. Ławryńczuk, Practical nonlinear predictive control algorithms for neural Wiener models, J. Process Control 23 (2013) 696–714.
[26] S. Mahmoodi, J. Poshtan, M.R. Jahed-Motlagh, A. Montazeri, Nonlinear model predictive control of a pH neutralization process based on Wiener–Laguerre model, Chem. Eng. J. 146 (2009) 328–337.
[27] A. Mozaffari, M. Vajedi, N.L. Azad, A robust safety-oriented autonomous cruise control scheme for electric vehicles based on model predictive control and online sequential extreme learning machine with a hyper-level fault tolerance-based supervisor, Neurocomputing 151 (2015) 845–856.
[28] O. Nelles, Nonlinear System Identification, From Classical Approaches to Neural Networks and Fuzzy Models, Springer, Berlin, 2001.
[29] J. Nocedal, S.J. Wright, Numerical Optimization, Springer, Berlin, 2006.
[30] S.J. Norquay, A. Palazoğlu, J.A. Romagnoli, Model predictive control based on Wiener models, Chem. Eng. Sci. 53 (1998) 75–84.
[31] S. Oblak, I. Škrjanc, Continuous-time Wiener-model predictive control of a pH process based on a PWL approximation, Chem. Eng. Sci. 65 (2010) 1720–1728.
[32] J. Peng, R. Dubay, J.M. Hernandez, M. Abu-Ayyad, A Wiener neural network-based identification and adaptive Generalized Predictive Control for nonlinear SISO systems, Ind. Eng. Chem. Res. 50 (2011) 7388–7397.
[33] P. Potočnik, I. Grabec, Nonlinear model predictive control of a cutting process, Neurocomputing 43 (2002) 107–126.
[34] J. Qiu, S. Ding, H. Gao, S. Yin, Fuzzy-model-based reliable static output feedback $\mathcal{H}_\infty$ control of nonlinear hyperbolic PDE systems, IEEE Trans. Fuzzy Syst. 24 (2016) 388–400.
[35] J. Qiu, G. Feng, H. Gao, Static-output-feedback $\mathcal{H}_\infty$ control of continuous-time T-S fuzzy affine systems via piecewise Lyapunov functions, IEEE Trans. Fuzzy Syst. 21 (2013) 245–261.
[36] M. Sarailoo, Z. Rahmani, B. Rezaie, A novel model predictive control scheme based on bees algorithm in a class of nonlinear systems: application to a three tank system, Neurocomputing 152 (2015) 294–304.
[37] B. Schölkopf, A. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT, Cambridge, 2001.
[38] J. Shin, H. Jin Kim, S. Park, Y. Kim, Model predictive flight control using adaptive support vector regression, Neurocomputing 73 (2010) 1031–1037.
[39] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific, Singapore, 2002.
[40] J.A.K. Suykens, J.D. Brabanter, L. Lukas, J. Vandewalle, Weighted least squares support vector machines: robustness and sparse approximation, Neurocomputing 48 (2002) 85–105.
[41] S. Syafiie, F. Tadeo, E. Martinez, Model-free learning control of neutralization processes using reinforcement learning, Eng. Appl. Artif. Intell. 20 (2007) 767–782.
[42] P. Tatjewski, Advanced Control of Industrial Processes, Structures and Algorithms, Springer, London, 2007.
[43] X. Tian, G. Chen, S. Chen, A data-based approach for multivariate model predictive control performance monitoring, Neurocomputing 74 (2011) 588–597.
[44] T. Wang, H. Gao, J. Qiu, A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control, IEEE Trans. Neural Netw. Learn. Syst. 27 (2016) 416–425.
[45] Q.C. Wang, J.Z. Zhang, Wiener model identification and nonlinear model predictive control of a pH neutralization process based on Laguerre filters and least squares support vector machines, J. Zhejiang Univ.-Sci. C (Comput. Electron.) 12 (2011) 25–35.
[46] X.-C. Xi, A.-N. Poo, S.-K. Chou, Support vector regression model predictive control on a HVAC plant, Control Eng. Pract. 15 (2007) 897–908.
[47] S.S. Yoon, T.W. Yoon, D.R. Yang, T.S. Kang, Indirect adaptive nonlinear control of a pH process, Comput. Chem. Eng. 26 (2002) 1223–1230.
[48] X.Y. Zeng, X.W. Chen, SMO-based pruning methods for sparse least-squares support vector machines, IEEE Trans. Neural Netw. 16 (2005) 1541–1546.

Maciej Ławryńczuk was born in Warsaw, Poland, in 1972. He obtained his M.Sc. in 1998, Ph.D. in 2003, D.Sc. in 2013, in automatic control, from Warsaw University of Technology, Faculty of Electronics and Information Technology. Since 2003 he has been employed at the same university at the Institute of Control and Computation Engineering: in 2003–2004 as a teaching assistant, in 2004–2015 as an assistant professor, since 2015 as an associate professor. He is the author or a co-author of 6 books and more than 100 other publications, including 30 journal articles. His research interests include advanced control algorithms, in particular MPC algorithms, set-point optimisation algorithms, soft computing methods, in particular neural networks, modelling and simulation.