15th Triennial World Congress, Barcelona, Spain

ADAPTIVE PREDICTIVE CONTROL OF A NEUTRALIZATION PLANT USING LOCAL MODEL NETWORKS

U. Halldorsson, A. Ali, H. Unbehauen and Chr. Schmid

Control Engineering Laboratory
Dep. of Electrical Engineering and Information Sciences,
Ruhr-Universität Bochum, D-44780 Bochum, Germany
Fax: ++49 234 32 14101
E-Mail: halldorsson@esr.ruhr-uni-bochum.de

Abstract: This contribution investigates the application of a nonlinear model based adaptive predictive control algorithm to a pH neutralization plant. A local linear model network is responsible for the online identification of the plant. The advantage of such a model is that the information needed by the predictive control algorithm to build gradients is directly available. This aspect reduces the computational cost significantly. Results of real-time control of a laboratory-scale plant are presented. *Copyright © 2002 IFAC*

Keywords: Nonlinear Predictive Control, Adaptive Control, pH Control, Local Linear Model Networks, Nonlinear Systems.

1. INTRODUCTION

During the past two decades various predictive control algorithms have been introduced and applied with great success. Most of these methods are based on linear system descriptions, allowing fast calculations and making the analytical analysis of the system easier to handle. The theory of linear model based predictive control has been intensively explored and is considered to be well understood (Clarke, 1994). Nevertheless, it is well known that in the absence of nonlinearities and constraints, the finite-horizon linear predictive control hardly performs any better than the infinite-horizon linear quadratic (LQ) control (Allgöwer and Zheng, 2000). This does not happen in the nonlinear case, since the infinite horizon optimization becomes computationally intractable. However having the goal of improving the control quality, it is desirable to work directly with an underlying nonlinear model, without simplifying it to one linear model alone. Such approaches include the use of adaptive linear models for prediction (Gambier, 1995) or a global nonlinear model of the plant (Foss *et al.*, 1995). The second approach should from the theoretical point of view deliver a better control performance than the first, but it complicates the predictive control theory in many aspects considerably. Usually a nonlinear optimization problem must be solved using numerical methods, leading to tedious calculations of gradients and requiring long sampling times (Pottmann and Seborg, 1997). It is therefore of major importance that gradients are easily extractable from the nonlinear model used for prediction. To generalize this property for a class of dynamical systems, a fixed structure must be used for the nonlinear model applied. Since the late nineteen eighties there has been a considerable interest in applying neural networks to nonlinear system identification. Many network paradigms, e.g. multiple layer perceptron (MLP), radial basis

function (RBF) networks etc. have proved themselves to be excellent nonlinear models. The main drawback of such models is that the network parameters have no direct correspondence to the physical system parameters. For example, if the parameters of the linearized system at the current operation point or the gradient of some objective function are to be calculated, then the gradient of this neural model has to be calculated, which may be computationally extensive and error-prone due to the inherent ripples of the RBF or MLP networks. The local linear model (LLM) networks are free from RBF-like ripples. Another major advantage of such networks, which is introduced in this contribution, is that the calculation of gradients is straightforward. In this work we use a special class of LLM networks called Rectangular Local Linear Model (RLLM) networks as a nonlinear model of the system to devise an adaptive predictive control scheme.

The next section starts with a short introduction to the control scheme. After that a brief review of RLLM networks and their use as nonlinear models will be presented. Then we will summarize the fundamentals of nonlinear predictive control based on solving an optimization problem, followed by a discussion on the prediction of system output and calculation of required gradients. The proposed control scheme was applied to a laboratory-scale pH-neutralization plant presented in section 3. We will conclude with some remarks.

2. THE CONTROL SCHEME

The aim of this contribution is to use an online system identification method based on RLLM networks to generate the model needed for a nonlinear model based predictive control scheme. Nonlinear predictive control algorithms are known to be advantageous when the process shows the presence of: strong nonlinearities, constraints in control and state signals, or nonminimum phase properties. It is important that the generated nonlinear model captures all these characteristics with a high precision. Furthermore, since the nonlinear predictive control scheme to be considered is based on a numerical optimization, the nonlinear model must be applied to calculate gradients needed for optimization. Ripples, not existing in the system to be identified but occurring in the generated nonlinear model must be avoided, since they can strongly affect the gradient calculations, possibly damaging or leading to instability of the optimization process. Taking these aspects into account, the RLLM network was selected for the identification purpose.

2.1 Nonlinear Modelling using RLLM Networks

A wide range of nonlinear multiple-input-multiple-output dynamical systems can be described by an equation of the following form (Narendra and Parthasarathy, 1990)

$$\begin{aligned}\boldsymbol{y}(k) = \; & \boldsymbol{f}[\boldsymbol{y}(k-1), ..., \boldsymbol{y}(k-n_y), \\ & \boldsymbol{u}(k-1), ..., \boldsymbol{u}(k-n_u)] + \boldsymbol{\epsilon}(k),\end{aligned} \quad (1)$$

where $\boldsymbol{y}$ is the vector of m outputs and $\boldsymbol{u}$ is the vector of r inputs of the system, $\boldsymbol{\epsilon}$ is a noise vector, $\boldsymbol{f}$ is a nonlinear function, k is the discrete time instant and n_u and n_y are the maximum lags of the input and output signals respectively. This model is known as NARX-model (Sjöberg, 1995). The deterministic nonlinear mapping $\boldsymbol{f}$ of Eq.(1) can be approximated by using a proper nonlinear function approximator. Artificial neural networks are practically convenient approximators for nonlinear functions. The idea behind the local linear model (LLM) networks is to decompose the input space of the nonlinear mapping into a number of operating regions. These regions are selected in such a way that the given mapping can be approximated by a linear function in each of the regions. These operating regions are sometimes also known as operating regimes (Hunt and Johansen, 1997). There are several possibilities for determining the shape and form of these regions. Our approach, RLLM network, decomposes the input space of the function into (hyper-) rectangular regions (Junge and Unbehauen, 1998*a*). As the emphasis is on nonlinear predictive control, this contribution gives only a short introduction of the RLLM networks. For a detailed description of these networks, previous publications of the authors like (Ali *et al.*, 2000), (Junge and Unbehauen, 1998*a*), (Junge and Unbehauen, 1998*b*) may be consulted. The construction of RLLM networks involves three entirely different layers. The first (input) layer is composed of source nodes whose number L_0 is equal to the dimension of the input space. The second layer, also called hidden layer of high enough dimension L_1, consists of local linear units that are connected directly to all of the nodes in the input layer. The output layer is composed of linear units whose number L_2 is equal to the dimension of output space. The hidden layer of the network implements a nonlinear mapping from $\Re^{L_0}$ to $\Re^{L_1}$. The output of a certain local linear neuron j in the hidden layer having its center $\boldsymbol{c}_j$ dimensions $\boldsymbol{s}_j$ and linear weights $\boldsymbol{w}_j$ along with a bias b_j for an input vector $\boldsymbol{\varphi}(k)$ can be calculated as

$$g_j(k) = \left(b_j + \boldsymbol{w}_j[\boldsymbol{\varphi}(k) - \boldsymbol{c}_j]\right) v_j(k), \quad (2)$$

where v_j is the validity function of the local model. This function attains values between 0 and 1 enabling a smooth transition between different

local models. In our approach the validity function is defined as

$$v_j(k) = \begin{cases} 1 & a_j(k) \leq 1-\beta \\ f_a & 1-\beta < a_j(k) < 1+\beta \\ 0 & a_j(k) \geq 1+\beta, \end{cases} \quad (3)$$

where 2β is the width of the region in which two neighboring neurons overlap and a_j can be considered as a measure of the internal activity of the jth neuron and is defined as

$$a_j(k) = \max_{1 \leq l \leq L_0} \frac{|\varphi_l(k) - c_{lj}|}{0.5 s_{lj}}. \quad (4)$$

The function f_a in Eq.(3) can be considered as an activation function. It may be any continuous function, which causes a smooth transition between 0 and 1. The mapping from $\Re^{L_1}$ to $\Re^{L_2}$ is linear. The ith output $\hat{y}_i$ of the network can be given by

$$\hat{y}_i(k) = \sum_{j \in L_1^{(i)}} g_j(k), \quad (5)$$

where $L_1^{(i)}$ is the set of indices of hidden layer neurons making their contribution to the ith output. The online learning algorithm developed for the training of the above network, in addition to determining the optimum number of local linear models in the hidden layer, estimates the parameters of each linear model using a recursive least squares algorithm. This learning algorithm can be started with any initial form of the lattice. It automatically splits a neuron into two neurons if the behavior of the neuron is not satisfactory according to a given criterion. On the other hand, if two neighboring neurons have nearly identical parameters, then in order to achieve a parsimonious structure these neurons are merged together to give a single neuron (Junge and Unbehauen, 1998*a*).

2.2 Nonlinear Predictive Control Description

The model predictive control, also known as the receding-horizon control, is based on a simple problem formulation, which is well suited to deal with nonlinearities and constraints. Assuming that a discrete-time mathematical model is used, the basic idea is to determine the control action $\boldsymbol{u}(k)$ at time $t = kT$, by using the sampling time T and solving a finite-horizon optimization problem over a time interval of $t \in [kT, (k+N)T]$. Applying the receding-horizon principle for the next time instant $t = (k+1)T$, a new control $\boldsymbol{u}(k+1)$ is found by solving a new optimization problem for the next time interval $t \in [(k+1)T, (k+N+1)T]$. In mathematical terms this method can be described using the time-invariant nonlinear model of the plant

$$\hat{\boldsymbol{y}}(k+1) = \hat{\boldsymbol{f}}[\boldsymbol{y}(k), ..., \boldsymbol{y}(k-n_y+1), \boldsymbol{u}(k), ..., \boldsymbol{u}(k-n_u+1)] \quad (6)$$

with the same notation as in Eq.(1), using $[\boldsymbol{y}(k), ..., \boldsymbol{y}(k-n_y+1), \boldsymbol{u}(k), ..., \boldsymbol{u}(k-n_u+1)]$ as initial values and assuming that $\hat{\boldsymbol{f}}$ is a twice differentiable nonlinear approximation of $\boldsymbol{f}$, delivering the one-step prediction $\hat{\boldsymbol{y}}(k+1)$. The noise vector $\boldsymbol{\epsilon}$ in Eq.(1) must be omitted in Eq.(6) for prediction purposes. Furthermore, the input and output vectors $\boldsymbol{u}(k)$, $\boldsymbol{y}(k)$ must satisfy the boundary constraints

$$\hat{\boldsymbol{y}}(k^*+1) \in Y, \quad \boldsymbol{u}(k^*) \in U, \quad \forall k^* \geq k, \quad (7)$$

where Y and U are compact sets of $\Re^m$ and $\Re^r$ respectively. To be able to describe the desired control behavior, a quadratic objective function to be minimized is defined as

$$J(k) = \frac{1}{2} \sum_{i=1}^{N} \boldsymbol{e}(k+i)^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{e}(k+i) + \frac{1}{2} \sum_{i=0}^{N-1} \boldsymbol{u}^*(k+i)^{\mathrm{T}} \boldsymbol{R} \boldsymbol{u}^*(k+i), \quad (8)$$

where the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ are both symmetric and positive definite, and the vector $\boldsymbol{e}(k+i) = \boldsymbol{w}(k+i) - \hat{\boldsymbol{y}}(k+i)$ represents the deviation between the desired $\boldsymbol{w}(k+i)$ and predicted $\hat{\boldsymbol{y}}(k+i)$ output vector at time $t = (k+i)T$. There are several possibilities to define $\boldsymbol{u}^*$ such as successive incremental changes $\boldsymbol{u}^*(k) = \boldsymbol{u}(k) - \boldsymbol{u}(k-1)$ or deviation of control $\boldsymbol{u}^*(k) = \boldsymbol{u}(k) - \boldsymbol{u}_0$ from the steady state value $\boldsymbol{u}_0$. In this work the second version is chosen. The weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ can be used to influence the speed of control and to set the significance of the r input and m output signals for the optimization.

It should be mentioned that the objective function in Eqs. (8) solely presents one out of many possible objective functions used for predictive control. For example, considering the case of a fixed set-point $\{\boldsymbol{w}(k^*) = \boldsymbol{w}(k^*+1) | \forall k^* \geq k\}$, and attempting to provide an analytical stability proof for a nonlinear predictive control, it might be necessary to append the objective function in Eq.(8) with a terminal cost term $\boldsymbol{e}^{\mathrm{T}}(k+N)\boldsymbol{L}\boldsymbol{e}(k+N)$ and to append the constraints in Eq.(7) with a terminal state condition $\boldsymbol{e}(k+N) \in \Omega$ (Chen and Allgöwer, 1998). These additional terms have been omitted in this paper to simplify the following calculations.

Since the predictive control strategy is based on the receding-horizon principle, the optimization problem must be repeatedly solved after each sam-

pling instant. This property allows to consider the control law as a nonlinear function $\boldsymbol{\gamma}(\cdot)$

$$\begin{aligned}\boldsymbol{u}(k) = \boldsymbol{\gamma}[&\boldsymbol{y}(k-1), ..., \boldsymbol{y}(k-n_y),\\ &\boldsymbol{u}(k-1), ..., \boldsymbol{u}(k-n_u+1),\\ &\boldsymbol{w}(k), ..., \boldsymbol{w}(k+N-1)],\end{aligned} \tag{9}$$

which performs the optimization task

$$\min_{\boldsymbol{u}(k),...,\boldsymbol{u}(k+N-1)} J(k) \tag{10}$$

and returns only the first element of the solution $[\boldsymbol{u}(k), \ldots, \boldsymbol{u}(k+N-1)] \in \Re^{r\times N}$ for control. Although the elements $[\boldsymbol{u}(k+1), \ldots, \boldsymbol{u}(k+N-1)]$ resulting from the solution are usually discarded for control purposes, they can be effectively used to initiate the optimization procedure taking place at time $t = (k+1)T$. This is of significant importance if the optimization is performed using a gradient method as discussed in the following section.

2.3 Prediction and Optimization

For prediction and optimization a model of the plant must be used. In our approach an RLLM network is applied for this purpose, which delivers an estimate $\hat{\boldsymbol{f}}$ of the unknown plant according to the NARX-model description of Eq.(1).

Let us define a sequence of suggested controller outputs

$$\bar{\boldsymbol{u}}(k) = [\,\boldsymbol{u}(k)^{\mathrm{T}} \quad \ldots \quad \boldsymbol{u}(k+N-1)^{\mathrm{T}}\,]^{\mathrm{T}} \tag{11}$$

In order to estimate the future response of the plant to this sequence, the neural network model output can be calculated recursively to get

$$\bar{\boldsymbol{y}}(k) = [\,\hat{\boldsymbol{y}}(k+1)^{\mathrm{T}} \quad \ldots \quad \hat{\boldsymbol{y}}(k+N)^{\mathrm{T}}\,]^{\mathrm{T}}\,. \tag{12}$$

Each element $\hat{\boldsymbol{y}}(l)$ of the above vector is given by the following relation

$$\hat{\boldsymbol{y}}(l) = \hat{\boldsymbol{f}}[\boldsymbol{\varphi}(l)], \tag{13}$$

where

$$\begin{aligned}\boldsymbol{\varphi}(l) = [&\tilde{\boldsymbol{y}}(l-1)^{\mathrm{T}} \ \ldots \ \tilde{\boldsymbol{y}}(l-n_y)^{\mathrm{T}}\\ &\boldsymbol{u}(l-1)^{\mathrm{T}} \ \ldots \ \boldsymbol{u}(l-n_u)^{\mathrm{T}}]^{\mathrm{T}}\end{aligned} \tag{14}$$

and

$$\tilde{\boldsymbol{y}}(i) = \begin{cases}\boldsymbol{y}(i) & i \le k\\ \hat{\boldsymbol{y}}(i) & \text{otherwise}\end{cases} \tag{15}$$

It is clear from the nature of local linear model networks that different local linear models may be activated successively during this prediction process. Figure 1 demonstrates this phenomenon for a first-order SISO system. The points A and B correspond to the lower and upper boundaries of the prediction horizon, passing through the local linear models L_7, L_8, L_{10}, L_4, L_5 and L_6. In addition to predicting the system output at instant l, the network also returns a parameter matrix

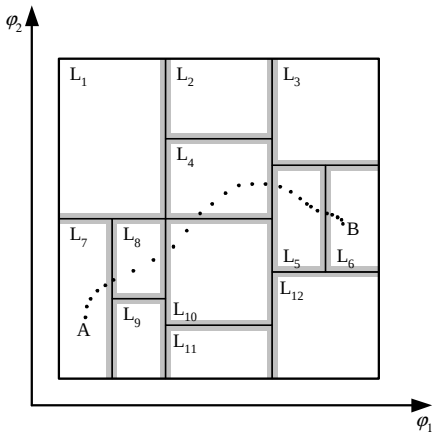

Fig. 1. Local Linear Model Regions

$$\begin{aligned}\boldsymbol{P}(l) = [&\boldsymbol{A}_1(l)\ \boldsymbol{A}_2(l)\ \ldots\ \boldsymbol{A}_{n_y}(l)\\ &\boldsymbol{B}_1(l)\ \boldsymbol{B}_2(l)\ \ldots\ \boldsymbol{B}_{n_u}(l)],\end{aligned} \tag{16}$$

which contains the parameters of the active local linear model

$$\begin{aligned}\hat{\boldsymbol{y}}(l) = &\sum_{i=1}^{n_y} \boldsymbol{A}_i(l)\tilde{\boldsymbol{y}}(l-i) +\\ &\sum_{i=1}^{n_u} \boldsymbol{B}_i(l)\boldsymbol{u}(l-i),\end{aligned} \tag{17}$$

where $\boldsymbol{A}_i \in \Re^{m\times m}$ and $\boldsymbol{B}_i \in \Re^{m\times r}$.

This expression can be simplified using the active input pattern vector $\boldsymbol{\varphi}(l)$ and the parameter matrix $\boldsymbol{P}(l)$, such that

$$\hat{\boldsymbol{y}}(l) = \boldsymbol{P}(l)\boldsymbol{\varphi}(l). \tag{18}$$

For calculating the control deviation, the desired reference signal sequence is assumed to be known over the prediction horizon

$$\bar{\boldsymbol{w}}(k) = [\,\boldsymbol{w}(k+1)^{\mathrm{T}} \quad \ldots \quad \boldsymbol{w}(k+N)^{\mathrm{T}}\,]^{\mathrm{T}} \tag{19}$$

This allows to define an error sequence

$$\bar{\boldsymbol{e}}(k) = [\,\boldsymbol{e}(k+1)^{\mathrm{T}} \quad \ldots \quad \boldsymbol{e}(k+N)^{\mathrm{T}}\,]^{\mathrm{T}} \tag{20}$$

where $\boldsymbol{e}(k+i) = \boldsymbol{w}(k+i) - \hat{\boldsymbol{y}}(k+i)$. Using this notation the objective function in Eq.(8) can be rewritten using $\bar{\boldsymbol{u}}^*(k) = \bar{\boldsymbol{u}}(k) - \bar{\boldsymbol{u}}_0$ as

$$\begin{aligned}J(k) = \frac{1}{2}[&\bar{\boldsymbol{e}}(k)^{\mathrm{T}}\bar{\boldsymbol{Q}}\bar{\boldsymbol{e}}(k) +\\ &\bar{\boldsymbol{u}}^*(k)^{\mathrm{T}}\bar{\boldsymbol{R}}\bar{\boldsymbol{u}}^*(k)],\end{aligned} \tag{21}$$

where

$$\bar{\boldsymbol{Q}} = \mathbf{I}_N \otimes \boldsymbol{Q} \qquad \bar{\boldsymbol{R}} = \mathbf{I}_N \otimes \boldsymbol{R}. \tag{22}$$

Here the operator $\otimes$ shows the Kronecker tensor product. To solve the optimization task described in Eq.(10) many numerical methods have been suggested in the literature (Himmelblau, 1972). These methods usually make use of a calculated gradient vector to approach the minimum iteratively. In this work the 'steepest descent' method is applied to update the solution according to

$$\bar{\boldsymbol{u}}(k)^{(n+1)} = \bar{\boldsymbol{u}}(k)^{(n)} - \operatorname{diag}(\boldsymbol{\lambda}(k)^{(n)}) \left.\frac{\delta J(k)}{\delta \bar{\boldsymbol{u}}(k)}\right|_{\bar{\boldsymbol{u}}(k)^{(n)}}, \quad (23)$$

where $\boldsymbol{\lambda}(k)^{(n)} \in \Re^{Nr}$ is a weighting vector used in order to improve the optimization speed. In this contribution an adaptive strategy was used to determine the weighting vector. A general treatment of such adaptive methods can be found in (Haykin, 1994). The gradient of the objective function above can be given as

$$\frac{\delta J(k)}{\delta \bar{\boldsymbol{u}}(k)} = \frac{\delta \bar{\boldsymbol{e}}(k)}{\delta \bar{\boldsymbol{u}}(k)} \bar{\boldsymbol{Q}} \bar{\boldsymbol{e}}(k) + \frac{\delta \bar{\boldsymbol{u}}^*(k)}{\delta \bar{\boldsymbol{u}}(k)} \bar{\boldsymbol{R}} \bar{\boldsymbol{u}}^*(k). \quad (24)$$

The signal vectors $\bar{\boldsymbol{e}}(k)$ and $\bar{\boldsymbol{u}}^*(k)$ are determined using Eqs. (20) and (11). The gradients of these vectors can be given as

$$\frac{\delta \bar{\boldsymbol{u}}^*(k)}{\delta \bar{\boldsymbol{u}}(k)} = \boldsymbol{I}_{Nn} \quad (25)$$

$$\frac{\delta \bar{\boldsymbol{e}}(k)}{\delta \bar{\boldsymbol{u}}(k)} = -\frac{\delta \bar{\boldsymbol{y}}(k)}{\delta \bar{\boldsymbol{u}}(k)} = -\boldsymbol{G}^{\mathrm{T}}(k). \quad (26)$$

The lower block triangular matrix $\{\boldsymbol{G}_{ij}\}$ is the Jacobian of $\bar{\boldsymbol{y}}(k)$ with respect to $\bar{\boldsymbol{u}}(k)$. The non-zero block elements of this matrix can be calculated recursively using parameter matrices $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$ according to the following scheme:

for $i = 1 : N$,
  for $j = 1 : i$,
    if $j > (i - n_u)$ then
      $\boldsymbol{G}_{ij}(k) = \boldsymbol{B}_{i-j+1}(k+i)$
    if $j < i$ then $\boldsymbol{G}_{ij}(k) = \boldsymbol{G}_{ij}(k) + \sum_{l=1}^{\min(n_y, i-1)} \boldsymbol{A}_l(k+i)\boldsymbol{G}_{i-l,j}(k)$
  end
end

The Hessian of the objective function can also be calculated in a similar way, which can be useful for other types of optimization methods.

3. EXPERIMENTAL RESULTS

The pH neutralization process is characterized by strong static and dynamic nonlinearities. The laboratory-scale neutralization plant to be controlled in this work is shown in Figure 2. The plant consists of a 5.5 liter stirred tank with two inlets and one outlet (Draeger *et al.*, 1995). Each of these inlets is connected to a reciprocating pump with adjustable sweep volume. The frequency of the pump is proportional to the input signal and can assume 180 uniformly distributed values. For the following experiment one pump was fed with 0.1 normal acetic acid (CH_3COOH) running at a fixed frequency and pumping 7.56 l/h. The second pump was driven by the control signal to inject 0.3 normal solution of the sodium hydroxide (NaOH) to bring $\mathrm{pH} = -\log[\mathrm{H}^+]$ of the tank contents at a desired level. The chemical reaction can be described as

$$\mathrm{Na^+OH^- + CH_3COO^-H^+} \Leftrightarrow \mathrm{Na^+CH_3COO^- + H_2O} \quad (27)$$

The control of such a plant has proven itself to be a great challenge. Many linear and nonlinear techniques have been developed and applied (Fikar and Draeger, 1995), (Gerksic and Juricic, 1999). The real-time adaptive predictive control results achieved during this work are shown in Figure 3. During this experiment the following settings were used for predictive control: $N = 8$, $\boldsymbol{R} = 0.125$ and $\boldsymbol{Q} = 1$. The system was sampled every 8 seconds. The experiment was started with one linear model of the second order. In the beginning the neural model was inaccurate and the control performance was not good. With the passage of time the learning algorithm produced 3 local linear models to cover the current range of operation. Figure 3 shows improvement in control performance as the network learned the nonlinear behavior of the process. To select the active linear model the delayed pH value pH(k-1) was chosen as scheduling variable. Operating range covered by each of these local models is shown in Figure 3 by thick dotted lines representing the boundaries. These three local models differ from each other in steady state as well as dynamic characteristics. As it is also clear from the figure that if the reference signal remains constant the system is operating in the vicinity of an operating point covered by

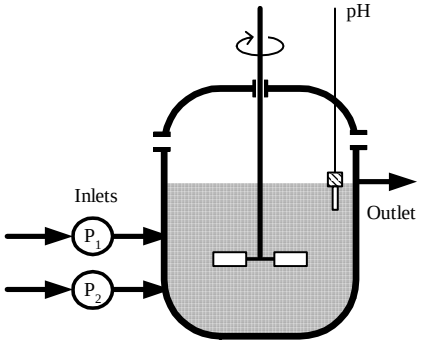

Fig. 2. The Neutralization Plant

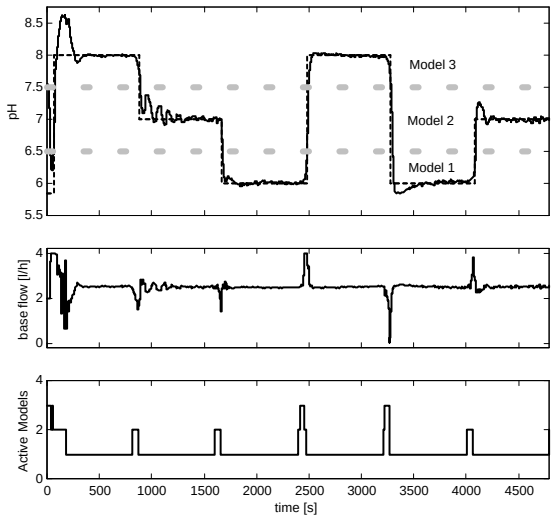

Fig. 3. Experimental Results

a single linear model. When the reference signal is suddenly changed from one point to the next, a single linear model is not able to predict the nonlinear behavior of the plant, a successive activation of different models during the prediction is the result. The last diagram of Figure 3 shows how many models are used at each time instant for prediction and calculation of gradients.

4. CONCLUSIONS

Adaptive nonlinear model-based predictive control using local linear model networks has been successfully implemented on a laboratory-scale pH-neutralization plant. The control scheme exploits the local linear property of individual models in the network in order to calculate gradients of the objective function efficiently. The model network is trained online to learn the unknown nonlinear dynamical behavior of the plant. The experimental control results confirmed the expected improvement in control as the neural network learned the nonlinear dynamics of the plant. Due to local linearity the calculation of gradients of the objective function is a simple task. Future research activities are focused on a continuous-time version of this scheme and stability analysis.

REFERENCES

Ali, A., M. Ashfaq and Chr. Schmid (2000). Model reference adaptive control of nonlinear systems using RLLM networks. *American Control Conference, Chicago* pp. 1659–1663.

Allgöwer, F. and A. Zheng (Eds.) (2000). *Nonlinear Model Predictive Control*. Vol. 26 of *Progress in Systems and Control Theory*. Birkhäuser. Basel.

Chen, H. and F. Allgöwer (1998). A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica* **34**(10), 1205–1217.

Clarke, D. (1994). *Advances in Model-Based Predictive Control*. Oxford University Press. Oxford.

Draeger, A., A. Engell and H. Ranke (1995). Model predictive control using neural networks. *IEEE Control Systems Magazine* **15**(5), 61–66.

Fikar, M. and A. Draeger (1995). Adaptive predictive control of a neutralization reactor. *The 10th Conference on Process Control, Tatranské Matliare, Slovakia* pp. 153–157.

Foss, B. A., T. A. Johansen and A. V. Sorensen (1995). Nonlinear predictive control using local models applied to a batch fermentation process. *Control Engineering Practice* **3**(3), 389 – 396.

Gambier, A. (1995). *State-Space design of predictive control for MIMO-Systems*. Ph.D. Thesis, Ruhr-Universität Bochum, Germany, Cuvillier Verlag. Göttingen, Germany.

Gerksic, S. and D. Juricic (1999). Wiener model based nonlinear predictive control of a pH neutralisation process. *IFAC Symposium on Dynamics and Control of Process, Corfu* pp. 547–552.

Haykin, S. (1994). *Neural Networks - A Comprehensive Foundation*. Macmillan College Publishing Company. New York.

Himmelblau, D. (1972). *Applied Nonlinear Programming*. McGraw-Hill Book Company. New York.

Hunt, K.J. and A. Johansen (1997). Design and analysis of gain-scheduled control using local controller networks. *International Journal of Control* **66**, 619–651.

Junge, T. and H. Unbehauen (1998*a*). Real-time control of a laboratory flight simulator by structurally adaptive RLLM networks. *Int. Conf. on Nonlinear Problems in Aviation and Aerospace, Daytona Beach* pp. 325–332.

Junge, T. and H. Unbehauen (1998*b*). Real-time learning control of an emergency turbogenerator plant using structurally adaptive rectangular local linear model networks. *Conf. of the IEEE Industrial Electronics Society, Aachen* pp. 2403–2408.

Narendra, K.S. and K. Parthasarathy (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on neural networks* **1**, 4–27.

Pottmann, M. and D.E. Seborg (1997). A nonlinear predictive control strategy based on radial basis function models. *Computers and Chemical Engineering* **21**(9), 965–980.

Sjöberg, J. (1995). *Nonlinear System Identification with Neural Networks*. Ph.D. Thesis, University of Linköping, Sweden.