Model Predictive Control Using Neural Networks
Andreas Draeger, Sebastian Engell, and Horst Ranke

In this article, we present the application of a neural-network-based model predictive control scheme to control pH in a laboratory-scale neutralization reactor. We use a feedforward neural network as the nonlinear prediction model in an extended DMC-algorithm to control the pH-value. The training data set for the neural network was obtained from measurements of the inputs and outputs of the real plant operating with a PI-controller. Thus, no a priori information about the dynamics of the plant and no special operating conditions of the plant were needed to design the controller. The training algorithm used is a combination of an adaptive backpropagation algorithm that tunes the connection weights with a genetic algorithm to modify the slopes of the activation function of each neuron. This combination turned out to be very robust against getting caught in local minima and it is very insensitive to the initial settings of the weights of the network. Experimental results show that the resulting control algorithm performs much better than the conventional PI-controller which was used for the generation of the training data set.

Introduction

In recent years, the requirements for the quality of automatic control in the process industries increased significantly due to the increased complexity of the plants and sharper specifications of product quality. At the same time, the available computing power increased to a very high level. As a result, computer models that are computationally expensive became applicable even to rather complex problems. Model-based control techniques were developed to obtain tighter control. Model predictive control was introduced successfully in several industrial plants. An important advantage of these control schemes is the ability to handle constraints of actuated variables and internal variables. In most applications of model predictive techniques, a linear model is used to predict the process behavior over the horizon of interest [1,2]. As most real processes show a nonlinear behavior, some work was done to extend predictive control techniques to incorporate nonlinear models [3-5].

The most expensive part of the realization of a nonlinear predictive control scheme is the derivation of the mathematical model. In many cases it is even impossible to obtain a suitable physically founded process model due to the complexity of the underlying processes or the lack of knowledge of critical parameters (as, e.g., temperature- and pressure-dependent mass transfer coefficients or viscosities) of the models. A promising way to overcome these problems is to use neural networks as nonlinear black-box models of the dynamic behavior of the process [6-8]. Such neural network models may be derived from measured input/output data of the plant. Usually, special open-loop experiments are performed to provide the data to train the neural nets. In many practical cases, however, conventional controllers (P/PID-controllers) are already in use at the plant which stabilize the plant and provide some basic, sometimes sluggish control. As we demonstrate, measurements of input/output variables of the plant operated with the linear controller may provide very good training data for the neural network. This approach is more practical (the plant is always under automatic control) and more effective (the excitation is much more similar to the one encountered under the final nonlinear control scheme) than using experiments without control (open-loop identification).

As a relatively well-known example, we consider a small-scale continuously operated neutralization reactor in which acetic acid is neutralized with sodium hydroxide solution. The plant exhibits a strongly nonlinear behavior, so that a nonlinear prediction model has to be used.

In the second section, we describe the plant in detail. The third section contains the neural network architecture considered here, the chosen training algorithm, and the generation of the training data set. In the fourth section, the extended DMC algorithm is described briefly with regard to its application with neural networks as prediction models. In the final section, we present the experimental results obtained at the real plant.

The Neutralization Reactor

The neutralization plant that we want to control consists of a continuous stirred tank with two inlets and one outlet [9] (see Fig. 1).

The hold-up of the tank is 5.57 l, and the concentrations of the acid and the sodium hydroxide solution streams are ≈0.01 mol/l. The acid flow rate was fixed at 0.33 l min⁻¹, whereas the sodium hydroxide flow is manipulated by the controller. In order to obtain the necessary precision of the flow rates, diaphragm pumps were chosen. All control actions are performed by a PC-based process control system which has a minimal sampling period of 1 s. The sodium hydroxide flow is controlled by the modulation of an impulse frequency which leads to a quantization of the control amplitude because the frequency can assume only certain discrete values.

In the tank, the acetic acid is neutralized with sodium hydroxide:

\[
Na^+OH^- + CH_3COO^- + e^- \rightarrow Na^+CH_3COO^- + H_2O.
\]
For diluted solutions, the pH-value can be obtained by:

\[ \text{pH} = - \log [H^+] \quad (2) \]

Due to the incomplete dissociation of acetic acid in water and the equilibrium reaction with sodium acetate, the system behaves like a buffer solution between pH 4 and pH 6.5. Consequently, the process gain varies extremely over the range of pH-values which can be controlled (see Fig. 2).

For the generation of input/output data of the plant, it was controlled by a PI-controller which was designed by pole-placement for pH 7 to get an acceptable compromise between setpoint tracking and disturbance rejection. For this design, the linearization of a physical model was used in our case. This controller could, however, also have been obtained by trial and error at the plant without modeling.

The Neural Network and Training Algorithm

Topology

For the prediction of the behavior of the neutralization reactor, we chose a feedforward network with sigmoid activation functions. This class of neural networks is well known and relatively well understood. Feedforward nets are easily implemented under real-time conditions. A disadvantage is that the training effort is usually high, which makes it difficult and time-consuming to explore various structures and to optimize the network structure. We overcame this problem, to a certain degree, by improving the training algorithms and by using several PCs in parallel in the training process.

Feedforward nets with at least one hidden layer have the capability to approximate any desired nonlinear mapping to an arbitrary degree of accuracy [10]. The neural net considered here consists of four layers: one input layer, two hidden layers, and one output layer. Even though networks with only one hidden layer already have the desired approximation power, our experience is that two hidden layers give better convergence in the training process. As inputs the actual and the last four old pH-values and the corresponding five values of the impulse frequency (which determines the sodium hydroxide flow) are fed into the network. The hidden layers consist of 10 neurons each, while the output layer consists of one neuron, the predicted next value of pH (see Fig. 3). The network thus performs a one-step-ahead prediction. In the predictive control scheme, however, it is used for a multi-step prediction by applying it recursively, i.e., past values of pH are replaced by predicted values. To use the neural net in this fashion requires a very good one-step-ahead prediction accuracy.

The network topology was chosen based on experiments with different structures. It is obvious that the net is redundant in the sense that from system-theoretic considerations, two past inputs should be sufficient because the order of the physical system is 2, at least in a first approximation, if the sensor dynamics are included. The results, however, were much better with more delayed inputs corresponding to a distribution of the information to more nodes than necessary. The same applies to the number of nodes in the hidden layers. The structure chosen is not minimal (and there is not much point in squeezing it to the limit), but the one that gave the best compromise in terms of robust prediction vs. training effort. The prediction error is not very sensitive to the number of neurons in the hidden layer.

Training Algorithm

In order to make the neural network perform the desired mapping from the input layer to the output layer, one usually searches for the optimal connection weights \( w \) between the neurons to approximate the desired mapping by so-called training algorithms. The most popular training algorithm for feedforward networks with sigmoid activation functions is the generalized delta-rule or backpropagation [11]:
where $E$ is the sum of the squares of the differences between network outputs $\Omega$ and the desired outputs (targets) $L$ for the set of $R$ training patterns:

$$E = \frac{1}{2} \sum_{n=1}^{R} (\Omega_n - L_n)^2.$$  \hspace{1cm} (4)

As the backpropagation algorithm is a steepest descent method, it has the disadvantage of converge very slowly and being vulnerable to getting caught in local minima of $E$. To overcome these disadvantages, a so-called momentum term can be included to slide over small minima in the error surface:

$$\Delta w_{ij}(t) = \kappa \cdot \frac{\partial E}{\partial w_{ij}} + \alpha \cdot \Delta w_{ij}(t-1).$$  \hspace{1cm} (5)

For further acceleration, the step size $\kappa$ can be chosen individually for each weight in the net and can also be adapted according to the learning progress [12]. All these improvements result in a significant speed-up of the learning process, but still a tendency to converge in local minima exists. The convergence properties of the algorithm are strongly dependent on the initial settings of the weights.

The backpropagation-based learning algorithm described above varies only the weights of the neural network to achieve the desired mapping. To overcome the dependence of the learning process on the initial settings and for a further improvement of the mapping accuracy, we use a combination of backpropagation and a genetic algorithm. The key idea is to vary the properties of each neuron in the net in addition to the adaptation of the weights. The processing properties of a neuron are determined by its activation function. We use a special type of sigmoid activation function, the so-called Fermi function:

$$S_f(x) = S_{\text{min}} + \frac{S_{\text{max}} - S_{\text{min}}}{1 + \exp\left(-\frac{4\sigma}{S_{\text{max}} - S_{\text{min}}} \frac{x}{S_{\text{max}} - S_{\text{min}}} \right)}.$$  \hspace{1cm} (6)

The genetic algorithm optimizes the slopes $\sigma$ of the Fermi functions (see Fig. 4). As the output of the network is essentially a sum over all weighted activation functions, we obtain a better accuracy by allowing to change the functions during training. The vector of slopes $\sigma$, in which the $i$th element denotes the slope of the Fermi function of neuron $i$, is treated as the string of genetic information. This string determines the mapping capability of the network, with respect to the overall modeling error $E$ computed from Equation (4). For each slope vector $\sigma$ an appropriate set of weights is determined by applying the improved backpropagation method SuperSAB [12] to the net. The training process thus is an iteration of the genetic algorithm which tunes the slopes and backpropagation to optimize the weights for each configuration of the slopes.

In the genetic algorithm, a population of neural nets, initialized with randomly distributed weights and slope values for all neurons, is considered. The genetic algorithm consists of three parts:

1. Determine the "fitness" of each member of the population. The member with the smallest value of $E$ is considered the fittest.
2. Generate a new population member by genetic operators such as mutation and recombination from old members. Train this new member a certain time with the SuperSAB method and determine the fitness of the new member.
3. If the new member is better than the worst old member, replace the worst old one with the new one.

Steps 1 to 3 are repeated until an acceptable performance by the best member is achieved.

By this combination of the genetic algorithm and a modified gradient-based method, the average training time to obtain a sufficiently small error is reduced considerably because ineffective training runs are avoided.

Training Data Generation

The aim of our work is to obtain a nonlinear black-box model of a real process by training a neural net from measured data only. Other researchers in the field who treated system identification with neural networks for a neutralization process generated the training data from applying random inputs to the process [13,14]. This leads to a very sparse representation of the high gain area at the point of equivalence in the training data set.

In contrast to this approach, we consider a training set which is generated by the system under closed-loop control with a reasonable PI-controller. Instead of generating random input sequences to the process, we changed the setpoint for the linear controller randomly during plant operation under control. This leads to a sequence of setpoint steps which were tracked by the conventionally controlled system. The resulting training set consists of the two measured time series of the pump frequency and pH-value for 3.5 hours for this conventionally tracked setpoint sequence (see Fig. 5).

The advantages of using data from the controlled plant go beyond the specific aspect of neutralization plants that without control the plant can hardly be operated within the most interesting region around pH 7. Also from the point of view of dynamic
excitation, it is desirable to be as close as possible to the inputs used for control during the identification experiment [15]. Another advantage of training the neural network with measured data generated by the controlled system is that suitable time series of data can be extracted from normal operation data (as long as sufficiently frequent and sufficiently large setpoint changes are possible). The only requirement is that the measurements represent the region of interest of the state space and that enough excitation relative to the unmeasured disturbances is provided.

The Extended DMC Algorithm

The Model Predictive Control Approach

In general, model predictive control is a strategy which is based on the explicit use of some kind of process model to predict the controlled variables over a certain time horizon, the prediction horizon. The control strategy consists of four parts:

1. At each sampling time, the value of the controlled variable $y(t+k)$ is predicted over the prediction horizon $k = 1 \ldots NP$. This prediction depends on the future values of the control variable $u(t+k)$ within a control horizon $k = 1 \ldots NU$, where $NU \leq NP$. If $NU < NP$ then $u(t+k) = u(t+NU)$, $k = NU+1 \ldots NP$.

2. A reference trajectory $r(t+k)$, $k = 1 \ldots NP$ is defined which describes the desired process trajectory over the prediction horizon.

3. The vector of future controls $u(t+k)$ is computed such that a cost function depending on the predicted control errors is minimized. The first element of the control vector is applied to the process.

4. The prediction error between the measured process output and the predicted output is used for disturbance estimation.

Steps 1 to 4 are repeated at each sampling instant; this is called a receding horizon strategy.

Extended DMC

We used the extended DMC algorithm described by Peterson et al. [3]. This algorithm is an extension of the linear DMC algorithm [1] to the explicit use of nonlinear models in the DMC scheme. In linear DMC, the prediction of the vector of future values of the controlled variables $y$ is performed using the dynamic matrix $A$, which consists of step response coefficients:

$$y_{lin} = y_{past} + A\Delta u + d$$  \hspace{1cm} (7)

In (7), the vector $d$ contains disturbances due to modeling errors as well as measurement errors.

The vector of the future control variables is obtained from the minimization of the following cost function:

$$f = \sum_{i=1}^{NP} \gamma(i) \left[ r(t+i) - y_{lin}(t+i) \right]^2 + \sum_{i=1}^{NU} \lambda(i) \Delta u(t+i)^2.$$  \hspace{1cm} (8)

The optimal control increments are computed as

$$\Delta u = (A^T \Gamma A + \Lambda^T \Lambda)^{-1} A^T \Gamma (r - y_{past} - d).$$  \hspace{1cm} (9)

where $\Gamma$, $\Lambda$ are diagonal matrices with the weighting parameters $\gamma$ resp. $\lambda$ on the diagonal.

As an extension to this linear algorithm, Peterson et al. proposed to split the disturbance vector in the linear model (7) such that one part represents the known nonlinearity of the process and the other the unknown influences:

$$d = d_{nl} + d^*,$$  \hspace{1cm} (10)

where

$$d^* = y - y_{lin}.$$  \hspace{1cm} (11)

$d_{nl}$ is computed using an iterative procedure until the predictions by the linear model (7) with the modified disturbance vector and by the nonlinear model coincide for the input vector which is computed by the linear DMC formula (9). This control vector is the solution of the extended DMC algorithm for the actual time step. The formula for the iterative computation is

$$d_{nl}^{i+1} = d_{nl}^i + \beta(y_{nl}^{i+1} - y_{lin}^{i+1}).$$  \hspace{1cm} (12)

where $y_{lin}$ is the prediction from the linear model (7) with the last value $d$ according to (10) and $\beta < 1$ is a relaxation parameter.

The resulting control scheme is shown in Fig. 6. In the original work by Peterson et al., the disturbance vector is assumed to be constant over the prediction horizon, where the modelling error $d^*$ is calculated from the prediction error of the nonlinear model for the first time interval. This produces a
steady-state offset if the model output for a steady-state situation is drifting away from the setpoint over the prediction horizon. As this was the case for our neural model (and it would be difficult to rule this behavior out in the training process), we estimate each element of the vector $d^*$ (i.e., the disturbances at instants $t+1$ to $t+NP$) individually. This was done by predicting the actual measured value from past pH-values with a delay of 1 to NP steps and using the difference of the prediction and the measurement at time $t$ to get an estimation of the error for $t+1$ to $t+NP$ [9]. So we obtain elements of $d^*$ for the whole prediction horizon. This more accurate calculation of the error leads to offset-free control and better disturbance rejection. However, in setpoint-tracking, the element-by-element calculation of $d^*$ may lead to an overreaction to past errors. Therefore we filter the vector $d^*$ according to

$$\Delta d^* = \frac{1}{T_d} (d_i^* - d_{i-1}^*),$$

(13)

### Experimental Results

For the experiments, the neural network shown in Fig. 3 was trained with data from the set shown in Fig. 5. The sampling time was chosen to be equal to the sampling time used for the PI-controller, which was $T_s = 5s$. Thus there were 2800 data points in the training set. The training algorithm described above was implemented in TurboPascal and Assembler on IBM-compatible PCs. The training was performed using 10 i486-based PCs running in parallel and connected via an Ethernet. The required training time was 4 hours. After the completion of the training, the prediction of the network was tested with data which was not included in the training set (see Fig. 7).

The extended DMC algorithm was realized using a commercial PC-based process monitoring and control system which accepts user-written PASCAL-programs. The prediction horizon was chosen as NP=5, whereas the control horizon was chosen as NU=1. The parameters $\theta(i)$ were all set to 1; $\lambda$ was set to 1.5. This corresponds to a rise time of roughly 25 s for a reference step. $T_d$ was set to 2 and the number of iterations of (12) was limited to 10. In contrast to the long training time, the time required to compute the network output online for the prediction and compute the controller output iteratively is short enough to be performed within one sampling period in addition to the standard tasks of the monitoring and control system on a PC. Fig. 8 shows the performance for setpoint tracking of the controller at the real plant compared to the carefully tuned linear fixed PI-controller.

The disturbance rejection performance of the controller was also tested (see Fig. 9). As a disturbance, a 20% decrease of the acid flow was performed at $t=0$, this flow rate was held constant afterwards. The peaks in the time series for the pH-value ($t=450$ and $t=1300$ respectively in Fig. 8 and $t=60s$ in Fig. 9) were caused by measurement errors. They do not represent real changes of the process, but were fed to both controllers without additional filtering. The performance obtained with the neural net plant model in the predictive control scheme is indeed very good and not the result of bad tuning of the PI-controller. A nonlinear control scheme derived from an analytical model of the process gave better results than the PI-controller [16] but also could not match the neural net based controller.

### Conclusions

We have presented an approach to model predictive control of a highly nonlinear plant without a priori information about the plant dynamics other than that the plant can be controlled (with unsatisfactory performance) by a conventional controller. The process
The controller was implemented using a small commercial PC-ries only. All we needed was data from 3.5 hours of normal plant operation under linear control with varying setpoints. No special input sequences as random input signals or steps were used.

We modified a nonlinear extension to the standard DMC control scheme for operation with the trained neural network. The controller was implemented using a small commercial PC-based process control system and shows a significantly improved performance compared to the linear controller for setpoint tracking as well as for disturbance rejection.

As we mentioned above, in contrast to the time needed for prediction, the training time for the neural network used here was quite long. Thus our research activities concentrate on networks with shorter training time for online training of time-varying process dynamics. As the DMC approach is inherently a multi-variable control strategy, we are also extending the proposed control algorithm to the MIMO case.

References


Andreas Draeger was born in 1965 in Dortmund, Germany. He received the Dipl.-Ing. degree in chemical engineering in 1991 from the University of Dortmund. Since 1991 he has been working as a research assistant towards his doctoral degree in the Process Control Group of the Chemical Engineering Department at that same university. His current research focuses on neural networks for system identification and predictive control.

Sebastian Engell was born in Düsseldorf, Germany, in 1954. He obtained the Dipl.-Ing. degree in electrical engineering from the University of Bochum in 1978, the Dr.-Ing. degree from the Department of Mechanical Engineering at the University of Duisburg in 1981, and the Venia Legendi in the same Department in 1987. He visited the Control Systems Centre at UMIST, Manchester, in 1978 and spent one year at McGill University, Montreal, in 1984-1985. From 1986 to 1990 he worked for the Fraunhofer-Institute IITB Karlsruhe, heading R&D groups in the area of process automation and control. Since 1990 he has been professor of process control at the Department of Chemical Engineering at the University of Dortmund. He is associate editor of the European Journal of Control and of Mathematical Modelling of Systems and co-editor of the IEEE Transactions on Control Systems Technology, and was elected to the board of the German Automation Society (GMA) for the period 1994-1996. His research interests are in controller design and on-line production scheduling.

Horst Ranke was born in 1966 in Harptstedt, Germany. He received the Dipl.-Ing. degree in chemical engineering in 1993 from the University of Dortmund.