Available online at www.sciencedirect.com
ScienceDirect
Journal of Process Control 16 (2006) 937–946
JOURNAL OF PROCESS CONTROL
www.elsevier.com/locate/jprocont

A neural network model predictive controller

Bernt M. Åkesson, Hannu T. Toivonen *

Department of Chemical Engineering, Åbo Akademi University, FIN-20500 Åbo, Finland

Received 1 November 2005; received in revised form 9 June 2006; accepted 11 June 2006

Abstract

A neural network controller is applied to the optimal model predictive control of constrained nonlinear systems. The control law is represented by a neural network function approximator, which is trained to minimize a control-relevant cost function. The proposed procedure can be applied to construct controllers with arbitrary structures, such as optimal reduced-order controllers and decentralized controllers.

Keywords: Model predictive control; Neural networks; Nonlinear control

1. Introduction

Due to the complexity of nonlinear control problems it is in general necessary to apply various computational or approximative procedures for their solution. In this context a widely used approach is model predictive control (MPC), which relies on solving a numerical optimization problem on line. Another approach is to apply function approximators, such as artificial neural networks, which can be trained off line to represent the optimal control law.

In model predictive control the control signal is determined by minimizing a future cost on-line at each time instant. It has found widespread industrial use for control of constrained multivariable systems and nonlinear processes [3,11,22]. A potential drawback of the MPC methodology is that the optimization problem may be computationally quite demanding, especially for nonlinear systems. In order to reduce the on-line computational requirements explicit model predictive control has been introduced for linear MPC problems [5], where part of the computations are performed off line.

The powerful function approximator properties of neural networks makes them useful for representing nonlinear models or controllers, cf. for example [4,13,14]. Methods based on model-following control have been particularly popular in neural network control. A limitation of this approach is, however, that it is not well suited for systems with unstable inverses. It is therefore well motivated to study methods based on optimal control techniques.

A number of neural network-based methods have been suggested for optimal control problems, where the control objective is to minimize a control-relevant cost function. One approach is to apply a neural network to approximate the solution of the dynamic programming equation associated with the optimal control problem [6]. A more direct approach is to mimic the MPC methodology and train a neural network controller in such a way that the future cost over a prediction horizon is minimized. One way to achieve this follows the explicit MPC technique, using a neural network to approximate a model predictive control strategy, which is mapped by off-line calculations [1,7,9,19]. Instead of training a neural network to approximate an optimal model predictive control strategy, an alternative and more direct approach is to train the neural network controller to minimize the cost directly, without the need to calculate a model predictive controller. Various versions of this procedure have been presented. Parisini

* Corresponding author. Tel.: +358 2 2154451; fax: +358 2 2154479.
E-mail addresses: bakesson@abo.fi (B.M. Åkesson), htoivone@abo.fi, Hannu.Toivonen@abo.fi (H.T. Toivonen).

0959-1524/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jprocont.2006.06.001

and Zoppoli [20] and Zoppoli et al. [24] study stochastic optimal control problems, where the controller is parameterized as a function of past inputs and outputs using neural network approximators. Seong and Widrow [23] consider an optimal control problem where the initial state has a random distribution, and the controller is a state feedback described by a neural network. In both studies a stochastic approximation type algorithm is used to train the network. Similar methods have been presented by Ahmed and Al-Dajani [2] and Nayeri et al. [15] using steepest descent methods to train the neural network controllers.

In many applications it is relevant to design controllers which have a specified structure. For complex systems it may for example be of interest to use a reduced-order controller, or to apply decentralized control for systems with many inputs and outputs. In model predictive control the complete system model is used to predict the future system trajectory, and the optimal control signal is therefore implicitly a function of the whole state of the system model. Hence, model predictive control is not applicable to fixed-structure control problems. In contrast, controllers based on neural network function approximators can be applied to optimal fixed-structure control by imposing the appropriate structure on the neural network approximator.

In this paper, the approach studied in [20,2,24,23,15] is formulated for constrained MPC type nonlinear optimal control problems with structural constraints. The control law is represented by a neural network approximator, which is trained off line to minimize a control-relevant cost function. The design of optimal low-order controllers is accomplished by specifying the input to the neural network controller appropriately. Decentralized and other fixed-structure neural network controllers are constructed by introducing appropriate constraints on the network structure. The performance of the neural network controller is evaluated on numerical examples and compared to optimal model predictive controllers.

2. Problem formulation

We consider the control of a discrete-time nonlinear system described by

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) \\ y(k) &= h(x(k)) \end{aligned} \tag{1}$$

where $y(k) \in R^p$ is the controlled output, $u(k) \in R^r$ is the manipulated input, and $x(k) \in R^n$ is a state vector. The control objective is to keep the output close to a specified reference trajectory $y_r(i)$ in such a way that large control signal variations are avoided and possible hard constraints on the states and inputs are satisfied. A commonly used, quantitative formulation of the control objective is to minimize the cost

$$\begin{aligned} J_N(U_N(k);k) = \sum_{i=k}^{k+N-1} \Big[&(\hat{y}(i+1|k) - y_r(i+1))^{\mathrm{T}} Q (\hat{y}(i+1|k) \\ &- y_r(i+1)) + \Delta u(i)^{\mathrm{T}} R \Delta u(i) \Big] + q_N(\hat{x}(k+N|k)) \end{aligned} \tag{2}$$

subject to the constraints

$$\begin{aligned} &g_x(\hat{x}(i+1|k)) \leqslant 0 \\ &g_u(u(i)) \leqslant 0 \\ &g_\Delta(\Delta u(i)) \leqslant 0, \quad i = k, k+1, \ldots, k+N-1 \end{aligned} \tag{3}$$

where $\hat{x}(\cdot \mid k)$ and $\hat{y}(\cdot|k)$ denote the predicted state and output as functions of future inputs and the state at time instant k. Here, $U_N(k)$ denotes the future inputs,

$$U_N(k) = \{u(k), u(k+1), \ldots, u(k+N-1)\} \tag{4}$$

and Δu denotes the incremental input,

$$\Delta u(k) = u(k) - u(k-1) \tag{5}$$

In Eq. (2), Q and R are symmetric positive (semi)definite output and input weight matrices, and $q_N(\cdot)$ is a non-negative terminal cost.

From the optimality principle of dynamic programming [6] it follows that minimization of the cost (2) gives the solution of the infinite-horizon optimal control problem obtained in the limiting case $N \to \infty$, if the terminal cost $q_N(\hat{x}(k+N \mid k))$ is taken as the minimum cost from time instant $k+N$ to infinity. The finite-horizon cost (2) is therefore well motivated. For nonlinear systems the optimal control problem has, however, in general no closed-form solution. Therefore, various brute-force and suboptimal methods have been studied.

In model predictive control (MPC) the control signal is determined by minimizing the cost (2) numerically with respect to the input sequence $U_N(k)$ at each sampling instant. Only the first element $u(k)$ of the optimal input sequence is applied to the system. In the next sampling period, a new optimization problem is solved to give the optimal control signal at sampling instant $k+1$, etc. In this approach the terminal cost $q_N(\cdot)$ can be taken as an approximation of the minimum cost from time instant $k+N$ to infinity, but is in practice more often used as a means to ensure closed-loop stability, or is omitted altogether if the optimization horizon is sufficiently long. The model predictive control approach has the drawback that a nonlinear, constrained optimization problem should be solved at each sampling period, which may be computationally too demanding for on-line implementation.

In order to reduce the computational burden in model predictive control, explicit MPC has been proposed. In this approach, part of the computations are performed off line. For nonlinear systems, the optimal MPC strategy may be mapped by off-line computations, and described by a function approximator. More precisely, the control strategy which minimizes the cost (2) defines the control signal, or equivalently, its increment $\Delta u(k)$, as a function,

$$\Delta u_{\text{opt}}(k) = g(I_{\text{MPC}}(k)) \tag{6}$$

where $I_{\text{MPC}}(k) = \{\hat{x}(k \mid k), y_r(k+1), y_r(k+2), \ldots, y_r(k+N)\}$ is the information used to compute the cost (2) as a function of $U_N(k)$. The functional relationship (6) can be evaluated for any $I_{\text{MPC}}(k)$ by minimizing the cost (2). The optimal strategy can therefore be approximated by a function approximator which can be trained off line. Although this approach has been found very useful, it has some limitations. As the MPC strategy is based on the information $I_{\text{MPC}}(k)$ used to calculate the predicted outputs, this approach is not well suited for representing reduced-order or fixed-structure controllers. In addition, the computational effort required to generate training data may be extensive, as each training data point requires the solution of an MPC optimization problem.

3. A neural network optimal controller

In this section a procedure for constructing a neural network model predictive controller for the control problem described in Section 2 is presented. Here we adopt a procedure in which the controller is trained directly to minimize the cost (2) for a training data set, without having to compute the optimal MPC control signals by off-line optimizations.

The controller is represented as

$$\Delta u(k) = g_{\text{NN}}(I(k); w) \tag{7}$$

where $g_{\text{NN}}(I(k); w)$ is a (neural network) function approximator, $I(k)$ denotes the information which is available to the controller at time instant k, and w denotes a vector of approximator parameters (neural network weights).

If complete state information is assumed, i.e., $I(k) = I_{\text{MPC}}(k)$, the controller (7) can be considered as a functional approximation of the optimal MPC strategy (6). The approach studied here is, however, not restricted to controllers with full state information, and typically the set $I(k)$ is taken to consist of a number of past inputs $u(k-i)$ and outputs $y(k-i)$ as well as information about the setpoint or reference trajectory $y_r(k+i)$. In this way it is possible to construct low-complexity controllers for high-order systems. Various ways to select the set $I(k)$ are illustrated in the examples presented in Section 4.

Remark 1. Besides allowing for controllers of reduced complexity the controller structure may be fixed as well by imposing a structure on the mapping $g_{\text{NN}}(\cdot;\cdot)$. For example, assuming that the information has the decomposition $I(k) = [I_1(k), I_2(k), \ldots, I_r(k)]$, a decentralized controller $\Delta u_i(k) = g_{\text{NN},i}(I_i(k); w_i)$, $i = 1, \ldots, r$ is obtained by requiring that the controller has the structure

$$g_{\text{NN}}(I(k); w) = \Big[g_{\text{NN},1}^{\text{T}}(I_1(k); w_1), g_{\text{NN},2}^{\text{T}}(I_2(k); w_2), \ldots, g_{\text{NN},r}^{\text{T}}(I_m(k); w_r)\Big]^{\text{T}} \tag{8}$$

In order to determine the controller parameters w in such a way that the control law (7) minimizes the cost (2) it is required that the cost is minimized for a set of training data,

$$V^{(m)}(k) = \big\{x^{(m)}(k), u^{(m)}(k-1), y_r^{(m)}(k+1), \ldots, y_r^{(m)}(k+N)\big\}, \quad m = 1, 2, \ldots, M \tag{9}$$

Using the control strategy (7), the system evolution for the initial state $x^{(m)}(k)$ is given by

$$\begin{aligned} x^{(m)}(i+1) &= f(x^{(m)}(i), u^{(m)}(i)) \\ \Delta u^{(m)}(i) &= g_{\text{NN}}(I(i); w) \\ y^{(m)}(i) &= h(x^{(m)}(i)), \quad i = k, k+1, \ldots \end{aligned} \tag{10}$$

Define the associated cost associated with the training data (9),

$$\begin{aligned} J_N^{(m)}(w) = \sum_{i=k}^{k+N-1} \Big[&(y^{(m)}(i+1) - y_r^{(m)}(i+1))^{\text{T}} Q (y^{(m)}(i+1) \\ &- y_r^{(m)}(i+1)) + \Delta u^{(m)}(i)^{\text{T}} R \Delta u^{(m)}(i)\Big] \\ &+ q_N(x^{(m)}(k+N)) \end{aligned} \tag{11}$$

The training of the approximator (7) now consists of solving the nonlinear least-squares optimization problem

$$\min_w \sum_{m=1}^{M} J_N^{(m)}(w) \tag{12}$$

subject to the constraints

$$\begin{aligned} &g_x(x^{(m)}(i+1)) \leqslant 0 \\ &g_u(u^{(m)}(i)) \leqslant 0 \\ &g_{\Delta}(\Delta u^{(m)}(i)) \leqslant 0, \quad i = k, k+1, \ldots, k+N-1 \end{aligned} \tag{13}$$

The training problem can be solved by a gradient-based nonlinear least-squares optimization procedure, such as the Levenberg–Marquardt algorithm. From Eq. (11), the cost function gradients are given by

$$\begin{aligned} \frac{\mathrm{d}J_N^{(m)}(w)}{\mathrm{d}w^{\text{T}}} = 2 \sum_{i=k}^{k+N-1} \Bigg(&(y^{(m)}(i+1) - y_r^{(m)}(i+1))^{\text{T}} Q \frac{\partial y^{(m)}(i+1)}{\partial w^{\text{T}}} \\ &+ \Delta u^{(m)}(i)^{\text{T}} R \frac{\partial \Delta u^{(m)}(i)}{\partial w^{\text{T}}} \Bigg) \end{aligned} \tag{14}$$

where

$$\begin{aligned} \frac{\partial y^{(m)}(i+1)}{\partial w^{\text{T}}} &= \frac{\partial h(x^{(m)}(i+1))}{\partial x^{(m)}(i+1)^{\text{T}}} \frac{\partial x^{(m)}(i+1)}{\partial w^{\text{T}}} \\ \frac{\partial \Delta u^{(m)}(i)}{\partial w^{\text{T}}} &= \frac{\partial g_{\text{NN}}(I(i); w)}{\partial w^{\text{T}}} + \frac{\partial g_{\text{NN}}(I(i); w)}{\partial x^{(m)}(i)^{\text{T}}} \frac{\partial x^{(m)}(i)}{\partial w^{\text{T}}} \end{aligned} \tag{15}$$

where $\partial g_{\text{NN}}(I(i); w)/\partial w^{\text{T}}$ is the partial derivative of the neural network output with respect to the network parameters, which depends on the network structure, and $\partial x^{(m)}(i)/\partial w^{\text{T}}$ is obtained recursively according to

$$\begin{aligned}
\frac{\partial x^{(m)}(i+1)}{\partial w^{\mathrm{T}}} &= \frac{\partial f(x^{(m)}(i),u^{(m)}(i))}{\partial x^{(m)}(i)^{\mathrm{T}}}\frac{\partial x^{(m)}(i)}{\partial w^{\mathrm{T}}} \\
&\quad + \frac{\partial f(x^{(m)}(i),u^{(m)}(i))}{\partial u^{(m)}(i)^{\mathrm{T}}}\frac{\partial u^{(m)}(i)}{\partial w^{\mathrm{T}}} \\
\frac{\partial u^{(m)}(i)}{\partial w^{\mathrm{T}}} &= \frac{\partial u^{(m)}(i-1)}{\partial w^{\mathrm{T}}} + \frac{\partial \Delta u^{(m)}(i)}{\partial w^{\mathrm{T}}} \\
\frac{\partial u^{(m)}(k-1)}{\partial w^{\mathrm{T}}} &= 0, \quad i = k, k+1, \ldots, k+N-1
\end{aligned} \tag{16}$$

Remark 2. The cost function (11) used to train the neural network controller is similar to the costs used in model predictive control. The proposed controller can therefore be regarded as an explicit model predictive controller. Notice that for a given controller complexity, the computational effort required to optimize the controller parameters w does not depend critically on the length N of the control horizon. It may therefore be feasible to use longer control horizons than in model predictive control, where the number of parameters to be optimized increases in proportion to length of the control horizon.

4. Numerical examples

In this section the neural network model predictive controller presented in Section 3 is illustrated on numerical examples. In all examples, the control law (7) is represented using a feedforward neural network with one hidden layer with hyperbolic tangent activation functions. This type of network can approximate any continuous nonlinear function to arbitrary accuracy [12]. The networks have the elements of $I(k)$ as inputs and $\Delta u(k)$ as outputs. The networks were trained using the Levenberg–Marquardt algorithm [21] to solve the nonlinear least-squares problem (12). The numerical computations were performed using the routine *lsqnonlin* of MATLAB's Optimization Toolbox.

In all examples, the optimization horizon N is taken sufficiently large for the closed-loop systems to reach steady state. Hence, a zero terminal cost is used in the cost function (11).

Example 1. In this example we consider the simulated pH neutralization process studied in [10,18,17,1]. In [1] an explicit MPC scheme was applied to the process by using a neural network to approximate the optimal MPC strategy. The process is described by nonlinear differential equations. By velocity-based linearization and integration over the sampling period, the process can be described by the state-dependent parameter model [17,1]

$$\begin{aligned}
x(k+1) &= F(y(k))x(k) + G_0(y(k))d(k) \\
&\quad + G_1(y(k))\Delta u(k-L) + G_v(y(k))v(k) \\
y(k) &= Hx(k) + n(k)
\end{aligned} \tag{17}$$

where the output $y(k)$ is the controlled pH value and $u(k)$ is the input flow used for control. The system is subject to a deterministic disturbance $d(k)$ and independent stochastic white noise disturbances $v(k)$ and $n(k)$ with variances R_v and R_n. Using the sampling time 0.2 min, the time delay is $L = 5$, and the system matrices can be represented as functions of the output according to

$$F(y) = \begin{bmatrix} p_1(y)+0.96 & p_2(y) & p_1(y)+0.96 & 0 \\ p_1(y) & p_2(y)+0.96 & p_1(y) & 0 \\ 0 & 0 & 0 & 0 \\ p_3(y) & p_4(y) & p_3(y) & 1 \end{bmatrix},$$

$$G_0(y) = 0.2\begin{bmatrix} p_1(y)+0.96 \\ p_1(y) \\ -1 \\ p_3(y) \end{bmatrix},$$

$$G_1(y) = 0.2\begin{bmatrix} p_2(y) \\ p_2(y)+0.96 \\ 0 \\ p_4(y) \end{bmatrix}, \quad G_v(y) = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

$$H = [\,0 \quad 0 \quad 0 \quad 1\,]$$

The state-dependent parameters are given by the expressions

$$\begin{aligned}
p_1(y) &= \frac{0.23}{1+\mathrm{e}^{-4.69y+24.59}} \\
p_2(y) &= \frac{-0.59}{1+\mathrm{e}^{-3.59y+19.14}} \\
p_3(y) &= -0.64\mathrm{e}^{-(y-4.58)^2/0.40} - 0.11 \\
p_4(y) &= 1.31\mathrm{e}^{-(y-4.53)^2/0.43} + 0.17
\end{aligned}$$

which are functional representations of the tabulated system parameters used in previous studies [1].

By Eq. (17), the predicted outputs $\hat{y}(k+1|k), \ldots, \hat{y}(k+L+N|k)$ can be determined according to

$$\hat{y}(k+l|k) = H\hat{x}(k+l \mid k), \quad l = 0, 1, \ldots, L+N \tag{18}$$

where the state estimates are given by the state-dependent Kalman filter [1]

$$\begin{aligned}
\hat{x}(k|k) &= \hat{x}(k|k-1) + K(k)[y(k) - H\hat{x}(k|k-1)] \\
\hat{x}(k+l+1|k) &= F(\hat{y}(k+l|k))\hat{x}(k+l|k) \\
&\quad + G_1(\hat{y}(k+l|k))\Delta u(k+l-L)
\end{aligned}$$

$l = 0, 1, \ldots, L+N$, where $K(k)$ is the Kalman filter gain, given by

$$\begin{aligned}
K(k) &= P(k)H^{\mathrm{T}}\left(HP(k)H^{\mathrm{T}} + R_n\right)^{-1} \\
P(k+1) &= F(\hat{y}(k \mid k))(I - K(k)H)P(k)F^{\mathrm{T}}(\hat{y}(k \mid k)) + G_v R_v G_v^{\mathrm{T}}
\end{aligned} \tag{19}$$

The optimal control strategy is a function of the predicted state $\hat{x}(k+L \mid k)$ and the reference trajectory $y_r(k+L+i)$, $i = 1, \ldots, N$. It will be assumed that the reference trajectory is given by a reference model

$$x_r(k+1) = F_r x_r(k) + G_r y_{sp}(k)$$
$$y_r(k) = H_r x_r(k) + J_r y_{sp}(k) \quad (20)$$

where $y_{sp}(k)$ denotes the setpoint, which is subject to step changes. It follows that the control strategy can be taken as a function of the predicted system state $\hat{x}(k+L|k)$, the state $x_r(k+L|k)$ of the reference model, and the setpoint $y_{sp}(k+L)$ at time instant $k+L$, i.e., the information $I(k)$ available to the controller (7) can be taken as

$$I(k) = \{\hat{x}(k+L|k), x_r(k+L), y_{sp}(k+L)\} \quad (21)$$

The design parameters were selected in accordance with [1]. The weights in the cost function (11) were $Q=1$ and $R=1$. The white noise variances $R_v = R_n = 0.001$ were used in the Kalman filter equation (19), and the reference model (20) was taken as a second-order system with unit stationary gain and a double pole at 0.9.

Due to the nonlinear system dynamics, rather large data sets are required in order to capture the optimal controller characteristics accurately in the whole operating region. This can be compared to nonlinear system identification, where long training sequences are also required [16]. The training data (9) were constructed so as to train the controller for both trajectory following and disturbance rejection in the output (pH) range $3 \leqslant y \leqslant 7$. Therefore two types of training data were generated. For trajectory following, training data were generated by taking the initial states as the stationary states corresponding to $y^{(m)}(l) = y_{sp}^{(m)}(l) = y_r^{(m)}(l) = y^{(m)}(k)$, $l < k$ and introducing the setpoint changes $y_{sp}^{(m)}(i) = y^{(m)}(k) \pm 1, i = k, k+1, \ldots$ Eight equally spaced initial pH values and fourteen setpoints were selected in the chosen pH range, giving a total of 14 reference trajectories. The control horizon in the cost function (11) was set to $N = 150$, which is sufficient to achieve a transition to the new setpoint according to the trajectory defined by the reference model (20). For disturbance rejection, training data were constructed by taking constant setpoints and reference signals $y_r^{(m)}(i) = y_{sp}^{(m)}(i), i = k, k+1, \ldots, k+N$ and initial states corresponding to steady states with $y^{(m)}(k) = y_{sp}^{(m)}(k) \pm 0.1$. The same fourteen different constant setpoints which were used for trajectory following were selected, resulting in a total of 28 initial states. In this case the control horizon taken as $N = 25$ steps, which corresponds to the time required for the systems to reach the setpoint after an initial offset. The total number of initial states and reference trajectories comprising the training set is thus $M = 42$. As each element of the training set contains N data points, the total number of data points is 2800.

The optimal network size was selected by using a separate test data set. The test data consisted of 36 sequences and 2400 data points, which were generated in a similar way as the training data, using six initial pH values in the range $3.5 \leqslant y \leqslant 6.5$ and setpoints in the range $3 \leqslant y_{sp} \leqslant 7$. Networks of different sizes were trained in order to find the one with the smallest test data cost. The training results are summarized in Table 1. The minimum value of the cost function on the test data is achieved using a network with 11 hidden layer nodes, and no further reduction could be obtained by increasing the network size. Therefore this network structure is used in the simulations below.

Table 1
Training results in Example 1 for networks with various numbers of hidden nodes (n_h)

	n_h	n_w	Cost	
			Training	Test
NN	5	41	4.03	3.21
NN	10	81	3.73	2.94
NN	11	89	3.71	2.93
NN	12	97	3.71	2.93
MPC	–	–	3.70	2.91

The total number of weights (n_w) is also given. Results obtained with the MPC strategy are included for comparison.

These results agree with those in [1], where a network of the same size was found to be optimal when used to approximate the MPC strategy. For comparison, Table 1 also shows the results obtained with a nonlinear MPC strategy. The prediction and control horizon of the MPC strategy was $N = 20$ steps.

In order to avoid convergence to local optima, the optimization of the neural network weights was performed using different initial points. In this example no problems with local optima were encountered. On average, the neural network weights converged in 400–500 iterations. This is comparable to the number of iterations required by a direct approximation of the optimal MPC strategy with a neural network of the same size [1] and using the same data points. However, as the trajectories (10) and associated cost (11) are evaluated at each iteration, the computational burden associated with the training is heavier than in the direct approximation. On the other hand, the direct approximation method requires the calculation of the optimal MPC control action for each data point, and its overall computational requirements are therefore heavier.

The closed-loop responses obtained with the neural network controller and the optimal MPC strategy for setpoint changes are given in Fig. 1. Fig. 2 shows the responses when there is also a white measurement noise with variance $R_n = 0.001$. Notice that the setpoint changes in Figs. 1 and 2 are distinct from the ones included in the training data. In order to further test the generalization properties of the neural network controller to situations not included in the training data, a sequence of setpoint changes was applied, where the changes take place before the new steady state has been reached. The responses in Fig. 3 show that the neural network controller performs well in this case as well.

Fig. 4 shows the responses to step disturbances in $d(k)$ at various steady-state pH values in the region 3–7. Step changes from 10 to 11 mmol/l occurred at time instant $k = 75$ and back to 10 mmol/l at time $k = 325$. Measurement noise was also present in the simulations. Notice that the disturbance $d(k)$ is unknown to the controller and the neural network controller had not been specifically trained

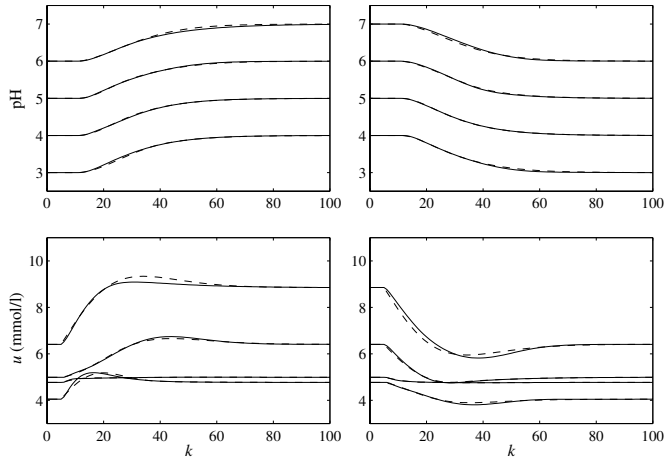

Fig. 1. Responses obtained in Example 1 with the neural network controller (solid lines) and model predictive control (dashed lines) for setpoint changes.

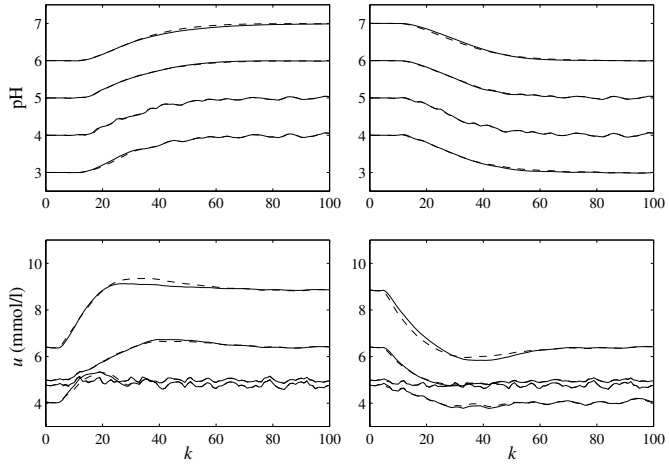

Fig. 2. Responses obtained in Example 1 with the neural network controller (solid lines) and model predictive control (dashed lines) for setpoint changes when there is measurement noise.

using step disturbances. Elimination of steady-state offsets was guaranteed by imposing the condition $\Delta u(k) = g_{\mathrm{NN}}(I(k); w) = 0$ at the steady states, cf. [1].

The simulation results show that the neural network controller achieves near-optimal control performance for various disturbance types. It should be noted that the neural network controller requires only 238 flops at each sampling interval, which is less than 0.1% of the average number of operations required by the model predictive controller. This is in accordance with the results in [1].

Example 2. In this example both centralized and decentralized neural network model predictive controllers are designed for a simulated multivariable non-isothermal continuous stirred tank reactor with a van de Vusse reaction scheme [8]. The process involves the reactant A, the desired product B, as well as two by-products C and D which are also produced in the reaction. The reactor is modelled by a system of four coupled differential equations,

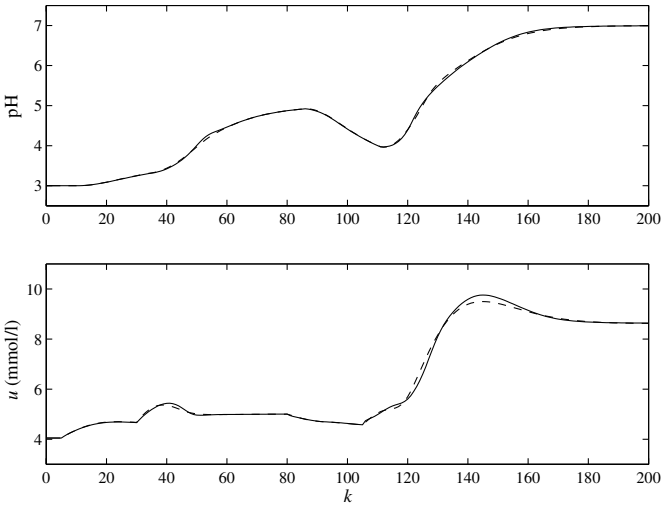

Fig. 3. Responses obtained in Example 1 with the neural network controller (solid lines) and model predictive control (dashed lines) for a sequence of setpoint changes. The total cost obtained when using the neural network controller is 2.60 and 1.60 when using the optimal MPC strategy.

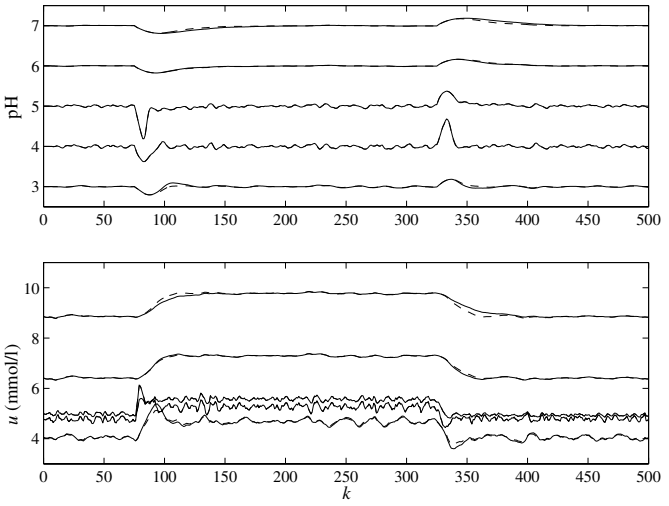

Fig. 4. Responses obtained in Example 1 with the neural network controller (solid lines) and model predictive control (dashed lines) for step disturbances when there is measurement noise. The total cost is 22.6 when using the neural network controller and 21.8 when using the optimal MPC strategy.

$$
\begin{aligned}
\frac{\mathrm{d}c_{\mathrm{A}}}{\mathrm{d}t} &= \frac{\dot{V}}{V_{\mathrm{R}}}(c_{\mathrm{A0}} - c_{\mathrm{A}}) - k_1(\vartheta)c_{\mathrm{A}} - k_3(\vartheta)c_{\mathrm{A}}^2 \\
\frac{\mathrm{d}c_{\mathrm{B}}}{\mathrm{d}t} &= -\frac{\dot{V}}{V_{\mathrm{R}}}c_{\mathrm{B}} + k_1(\vartheta)c_{\mathrm{A}} - k_2(\vartheta)c_{\mathrm{B}} \\
\frac{\mathrm{d}\vartheta}{\mathrm{d}t} &= -\frac{1}{\rho C_p}\left(k_1(\vartheta)c_{\mathrm{A}}\Delta H_{R_{\mathrm{AB}}} + k_2(\vartheta)c_{\mathrm{B}}\Delta H_{R_{\mathrm{BC}}} + k_3(\vartheta)c_{\mathrm{A}}^2\Delta H_{R_{\mathrm{AD}}}\right) \\
&\quad + \frac{\dot{V}}{V_{\mathrm{R}}}(\vartheta_0 - \vartheta) + \frac{k_{\mathrm{w}}A_{\mathrm{R}}}{\rho C_p V_{\mathrm{R}}}(\vartheta_{\mathrm{K}} - \vartheta) \\
\frac{\mathrm{d}\vartheta_{\mathrm{K}}}{\mathrm{d}t} &= \frac{1}{m_{\mathrm{K}}C_{p_{\mathrm{K}}}}\left(\dot{Q}_{\mathrm{K}} + k_{\mathrm{w}}A_{\mathrm{R}}(\vartheta - \vartheta_{\mathrm{K}})\right)
\end{aligned}
\tag{22}
$$

where c_A and c_B are the concentrations of components A and B, respectively, ϑ is the reactor temperature and ϑ_K is the coolant temperature. The concentration of A in the feed stream is c_{A0} and the temperature of the feed stream is ϑ_0. The reactor volume is V_R, $\dot{V}$ is the feed flow rate and $\dot{Q}_K$ is the rate of heat addition or removal. The reaction coefficients k_1, k_2 and k_3 are given by the Arrhenius equation,

$$k_i(\vartheta) = k_{0i} \cdot \exp\left(\frac{E_{Ai}}{\vartheta + 273.15\ ^\circ\mathrm{C}}\right), \quad i = 1, 2, 3 \tag{23}$$

Numerical values of the model parameters are given in Table 2.

The control objective is to control the concentration c_B and the reactor temperature ϑ by manipulating the feed flow rate $\dot{V}$ and the rate of heat addition or removal $\dot{Q}_K$. The concentration c_B and the reactor temperature ϑ are available from measurements. The feed concentration c_{A0} and the feed temperature ϑ_0 are treated as disturbances, with the nominal values 5.1 mol/l and 130 °C, respectively. The feed flow $\dot{V}$ is constrained to the interval $0.05\ \mathrm{m^3/h} \leqslant \dot{V} \leqslant 0.35\ \mathrm{m^3/h}$ and the rate of heat removal $\dot{Q}_K$ lies in the range $-9000\ \mathrm{kJ/h} \leqslant \dot{Q}_K \leqslant 0\ \mathrm{kJ/h}$.

Table 2
Parameters of the van de Vusse reactor

$k_{01} = 1.287 \times 10^{12}\ \mathrm{h^{-1}}$	$\Delta H_{R_{AB}} = 4.2\ \mathrm{kJ/mol}$
$k_{02} = 1.287 \times 10^{12}\ \mathrm{h^{-1}}$	$\Delta H_{R_{BC}} = -11.0\ \mathrm{kJ/mol}$
$k_{03} = 9.043 \times 10^{9}\ \mathrm{l/(mol\ h)}$	$\Delta H_{R_{AD}} = -41.85\ \mathrm{kJ/mol}$
$E_{A1} = -9758.3\ \mathrm{K}$	$\rho = 0.9342\ \mathrm{kg/l}$
$E_{A2} = -9758.3\ \mathrm{K}$	$C_p = 3.01\ \mathrm{kJ/(kg\ K)}$
$E_{A3} = -8560\ \mathrm{K}$	$k_w = 4032\ \mathrm{kJ/(h\ m^2\ K)}$
$A_R = 0.215\ \mathrm{m^2}$	$V_R = 0.01\ \mathrm{m^3}$
$m_K = 5.0\ \mathrm{kg}$	$C_{p_K} = 2.0\ \mathrm{kJ/(kg\ K)}$

A discrete-time system representation of the form (1) was constructed by using Euler's approximation to discretize the model (22) with the sampling time 20 s. The controlled outputs were taken as $y_1 = c_B$ [mol/l] and $y_2 = \vartheta$ [°C] and the inputs were defined as $u_1 = \dot{V}$ [m^3/h] and $u_2 = \dot{Q}_K$ [kJ/h].

The procedure in Section 3 was used to design both centralized and decentralized neural network controllers for the reactor system. Training data were generated in analogy with Example 1, with the outputs and setpoints in the region $0.8\ \mathrm{mol/l} \leqslant c_B \leqslant 1.0\ \mathrm{mol/l}$, $125\ ^\circ\mathrm{C} \leqslant \vartheta \leqslant 135\ ^\circ\mathrm{C}$. For trajectory following, the initial states were taken to correspond to steady states associated with the outputs $y_1^m(k), y_2^m(k)$ and setpoints $y_{sp,1}^m$, $y_{sp,2}^m$. Four equally spaced values in the considered regions were selected for each output. For each initial steady state, eight combinations of setpoint values $(y_{sp,1}^m, y_{sp,2}^m) = (y_1^m(k) \pm 0.1, y_2^m(k) \pm 5), (y_1^m(k), y_2^m(k) \pm 5), (y_1^m(k) \pm 0.1, y_2^m(k))$ were selected. Excluding values falling outside the selected variable range, this results in 84 initial states. The length of the prediction horizon was set to $N = 60$ steps. For disturbance rejection, training data were constructed by taking constant reference signals $y_{r1}^m(i) = y_{sp,1}^m, y_{r2}^m(i) = y_{sp,2}^m, i = k, k+1, \ldots$ and initial states corresponding to steady states with $y_1^m(k) = y_{sp,1}^m \pm 0.02$ and $y_2^m(k) = y_{sp,2}^m \pm 1$. Using four equally spaced values for each setpoint, 64 initial states are obtained. The length of the prediction horizon was $N = 25$. Thus, the total number of initial states in the training set was $M = 148$, and the number of training data points was 6640. The test data set consisted of four setpoint changes between the values $c_B = 0.85$ and 0.95 mol/l and the values $\vartheta = 128$ and 133 °C (cf. Figs. 5–7), and was used as in Example 1 to select the network sizes.

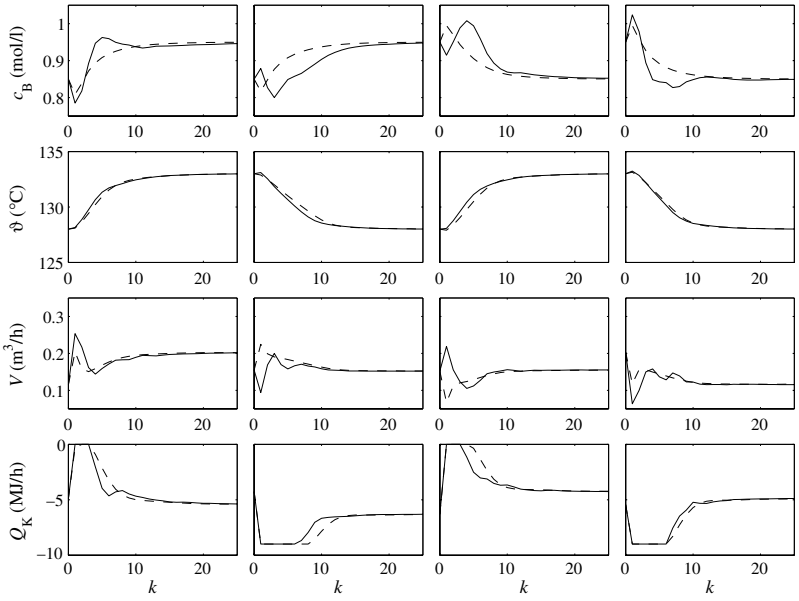

Fig. 5. Responses obtained in Example 2 to four setpoint changes when using a neural network controller (solid lines) and an optimal model predictive controller (dashed lines) designed for the weights in Eq. (25).

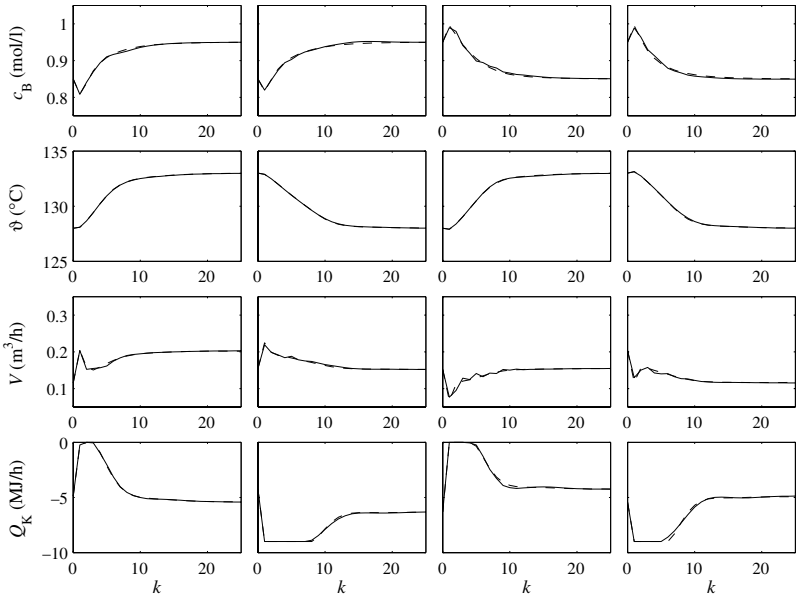

Fig. 6. Responses obtained in Example 2 to four setpoint changes used as test data when using a centralized neural network controller (solid lines) and an optimal model predictive controller (dashed lines) designed for the weights in Eq. (26).

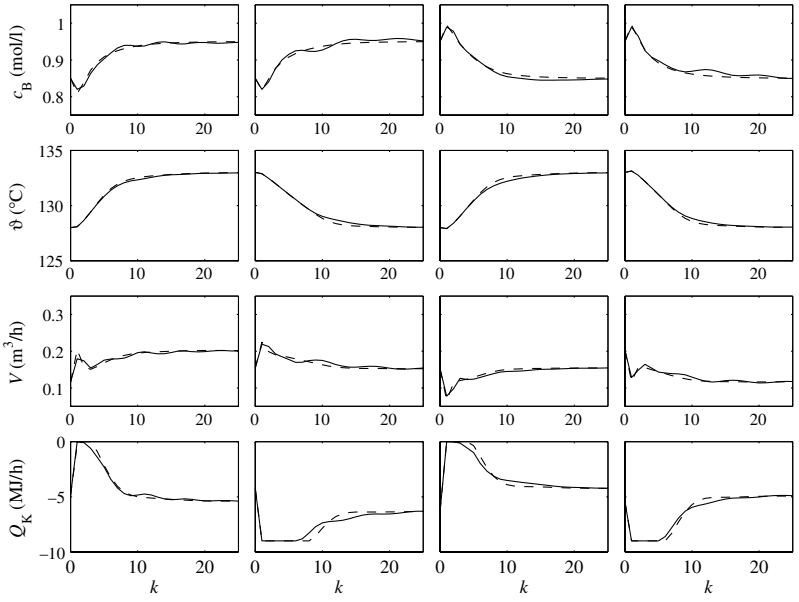

Fig. 7. Responses obtained in Example 2 to four setpoint changes used as test data when using a decentralized neural network controller (solid lines) and an optimal model predictive controller (dashed lines).

The neural network controllers were taken to be functions of past outputs $y(k-i)$ and input increments $\Delta u(k-i)$, the state $x_r(k)$ of the reference model (cf. Eq. (20)) and the current setpoint. Both outputs had identical reference models, which were taken as first-order systems with unit stationary gains and poles at 0.8. In the centralized controller case the controller in Eq. (7) was taken as a function of

$$I(k) = \{y(k), \ldots, y(k-n_y+1), \Delta u(k-1), \ldots, \Delta u(k-n_u), x_r(k), y_{sp}(k)\} \quad (24)$$

where $n_y = 3$ and $n_u = 3$. No significant improvement could be obtained by increasing the values n_y and n_u.

For the multivariable example system, the training of the neural network controllers turned out to be more demanding than for the single-input single-output system studied in Example 1. In particular, the optimization of the neural network weights could be stuck in local minima. These were avoided by starting the optimization from a number of random initial points.

It was observed that in the multivariable case, the choice of the relative magnitudes of the weights on the outputs in

the cost function (11) has a significant effect on the network training performance. If the contribution to the cost function from one output dominates over the contributions from the other outputs, the network will only learn to control the dominating output, whereas satisfactory control of the other outputs is not achieved. This happens despite the fact that the optimal model predictive controller based on the same weights achieves good control performance for all outputs. For the inputs a similar situation holds. The set of cost function weights for which a neural network controller can be efficiently trained by the procedure in Section 3 is therefore limited. This behaviour is illustrated in Fig. 5, which shows the closed-loop responses obtained with the optimal MPC strategy and the best neural network approximator which could be found when the weight matrices in the cost function (11) were taken as

$$Q = \begin{bmatrix} 400 & 0 \\ 0 & 10 \end{bmatrix}, \quad R = \begin{bmatrix} 10 & 0 \\ 0 & 10^{-7} \end{bmatrix} \tag{25}$$

The contribution of the first output c_B to the cost is much smaller than the contribution of the second output ϑ. Consequently, the second output dominates and poor control of the first output is obtained. Therefore, the weight on the first output should be increased. A similar phenomenon, although less prominent, can be seen for the inputs.

The controllers considered below will the based on the weight matrices

$$Q = \begin{bmatrix} 2500 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} 90 & 0 \\ 0 & 10^{-7} \end{bmatrix} \tag{26}$$

For these weights, the contributions to the cost from the individual outputs (inputs) will have the same orders of magnitude when using the optimal strategy.

The best performance on the test data when using a controller with the input in Eq. (24) was achieved with a network having four hidden layer neurons, and 78 weights. The cost on the training data was 2336 and on the test data the cost was 128.3. For comparison, with an optimal model predictive controller the training data cost was 1719 and the test data cost was 125.8. The responses of the neural network controller and model predictive control on the test data sequences are shown in Fig. 6. Notice that the system has inverse response characteristics, and both controllers give inverse responses.

An optimal decentralized neural network controller was designed as follows. For the chemical reactor it is natural to use a decentralized controller where the feed flow u_1 is a function of the product concentration y_1 and the rate of heat exchange u_2 is a function of the reactor temperature y_2. Hence a decentralized neural network controller consisting of the individual controllers

$$\Delta u_i(k) = g_{\mathrm{NN},i}(I_i(k); w_i), \quad i = 1, 2 \tag{27}$$

was used, where the information available to the individual controllers consisted of local variables only,

$$I_i(k) = \{y_i(k), \ldots, y_i(k - n_{y,i} + 1), \Delta u_i(k-1), \ldots, \Delta u_i(k - n_{u,i}), x_{r,i}(k), y_{\mathrm{sp},i}(k)\}, \quad i = 1, 2 \tag{28}$$

where $n_{y,i} = n_{u,i} = 3$, $i = 1, 2$. No significant improvement could be obtained by increasing the values $n_{y,i}$ and $n_{u,i}$. The network sizes were determined as above. A neural network controller with four hidden layer neurons was determined to control the concentration, while a network with three hidden layer neurons was used to control the temperature. The total number of weights is 72. The network gives the cost 3145 on the training data and 139.2 on the test data. Fig. 7 gives the closed-loop responses for the test data, showing that the performance of the decentralized controller is remarkably good for setpoint changes, despite the facts that there are considerable interactions between the loops (cf. Eq. (22)).

It should be observed that although the decentralized controllers are represented by separate networks, they cannot be trained independently. This is because due to the interactions between the loops one controller affects the output controlled by the other and vice versa. The simultaneous training of the networks can be performed in a straightforward way by defining the global parameter vector $w = [w_1^T, w_2^T]^T$ and by applying a standard gradient-based nonlinear least-squares approach of the form described in Section 3. The training is further complicated by the fact that the information $I_i(k)$ used to determine the control variable $u_i(k)$ does not define the state of the system uniquely. For these reasons the training of the decentralized neural network controllers proved to be more demanding both computationally and with respect to the quality of training data required. In this example, the whole training data set was essential in order to train the decentralized controller properly, whereas the centralized controller could be trained satisfactorily even though the number of data points were reduced.

5. Conclusion

Optimal neural network control of constrained nonlinear systems has been studied. The neural network controller is designed by minimizing an MPC type cost function off-line for a set of training data. In this way the procedure is closely related to MPC, and it can be considered as a form of explicit model predictive control.

The neural network controller has a number of distinct advantages over standard nonlinear model predictive control. In analogy with other explicit MPC methods, the neural network controller has substantially reduced on-line computational requirements. In addition, the computational effort involved in the network training depends mainly on the network complexity, and not on the length of the control horizon. This makes it feasible to design controllers with a longer control horizon than might be possible in MPC. Moreover, the structure of the neural network controller can be fixed, so that controllers with a

specified structure, such as decentralized controllers, can be designed. The main limitation of the neural network controller is that substantial off-line computations may be needed in order to train it properly, and for some choices of cost functions it may not even be feasible to achieve satisfactory accuracies.

Numerical examples show that the neural network model predictive controller can be trained to achieve near-optimal control performance (when compared to the optimal MPC strategy) using both centralized and decentralized controller structures.

Acknowledgement

This work was supported by the Academy of Finland (Grant 206750). Bernt M. Åkesson was supported by the Finnish Graduate School in Chemical Engineering (GSCE).

References

[1] 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, Computers & Chemical Engineering 29 (2) (2005) 323–335.
[2] M.S. Ahmed, M.A. Al-Dajani, Neural regulator design, Neural Networks 11 (9) (1998) 1695–1709.
[3] F. Allgöwer, T.A. Badgwell, S.J. Qin, J.B. Rawlings, S.J. Wright, Nonlinear predictive control and moving horizon estimation – an introductory overview, in: P.M. Frank (Ed.), Advances in Control: Highlights of ECC'99, Springer-Verlag, Berlin, 1999, pp. 391–449 (Chapter 12).
[4] S.N. Balakrishnan, R.D. Weil, Neurocontrol: a literature survey, Mathematical and Computer Modelling 23 (1–2) (1996) 101–117.
[5] A. Bemporad, M. Morari, V. Dua, E.N. Pistikopoulos, The explicit linear quadratic regulator for constrained systems, Automatica 38 (1) (2002) 3–20.
[6] D.P. Bertsekas, J.N. Tsitsiklis, Neuro-Dynamic Programming, Athena Scientific, Belmont, Mass, 1996.
[7] L. Cavagnari, L. Magni, R. Scattolini, Neural network implementation of nonlinear receding-horizon control, Neural Computing & Applications 8 (1) (1999) 86–92.
[8] S. Engell, K.-U. Klatt, Nonlinear control of a non-minimum phase CSTR, in: Proceedings of the American Control Conference, San Francisco, CA, USA, 1993, pp. 2941–2945.
[9] J. Gómez Ortega, E.F. Camacho, Mobile robot navigation in a partially structured static environment, using neural predictive control, Control Engineering Practice 4 (12) (1996) 1669–1679.
[10] T.K. Gustafsson, B.O. Skrifvars, K.V. Sandström, K.V. Waller, Modeling of pH for control, Industrial & Engineering Chemistry Research 34 (3) (1995) 820–827.
[11] M.A. Henson, Non-linear model predictive control: Current status and future directions, Computers & Chemical Engineering 23 (2) (1998) 187–202.
[12] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366.
[13] K.J. Hunt, D. Sbarbaro, R. Zbikowski, P.J. Gawthrop, Neural networks for control—a survey, Automatica 28 (6) (1992) 1083–1112.
[14] M.A. Hussain, Review of the applications of neural networks in chemical process control – simulation and online implementation, Artificial Intelligence in Engineering 13 (1) (1999) 55–68.
[15] M.R.D. Nayeri, A. Alasty, K. Daneshjou, Neural optimal control of flexible spacecraft slew maneuver, Acta Astronautica 55 (10) (2004) 817–827.
[16] M. Nørgaard, O. Ravn, N.K. Poulsen, L.K. Hansen, Neural Networks for Modelling and Control of Dynamic Systems, Springer-Verlag, London, 2000.
[17] R.H. Nyström, B.M. Åkesson, H.T. Toivonen, Gain-scheduling controllers based on velocity-form linear parameter-varying models applied to an example process, Industrial & Engineering Chemistry Research 41 (2) (2002) 220–229.
[18] R.H. Nyström, K.V. Sandström, T.K. Gustafsson, H.T. Toivonen, Multimodel robust control of nonlinear plants: a case study, Journal of Process Control 9 (2) (1999) 135–150.
[19] T. Parisini, R. Zoppoli, A receding-horizon regulator for nonlinear systems and neural approximation, Automatica 31 (10) (1995) 1443–1451.
[20] T. Parisini, R. Zoppoli, Neural approximations for multistage optimal control of nonlinear stochastic systems, IEEE Transactions on Automatic Control 41 (6) (1996) 889–895.
[21] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, 1992.
[22] S.J. Qin, T.A. Badgwell, An overview of nonlinear model predictive control applications, in: F. Allgöwer, A. Zheng (Eds.), Nonlinear Model Predictive Control, Birkhäuser Verlag, Basel, 2000.
[23] C.-Y. Seong, B. Widrow, Neural dynamic optimization for control systems—Part II: Theory, IEEE Transactions on Systems, Man, and Cybernetics 31 (4) (2001) 490–501.
[24] R. Zoppoli, M. Sanguineti, T. Parisini, Can we cope with the curse of dimensionality in optimal control by using neural approximators? in: Proceedings of the 40th IEEE Conference on Decision and Control, Orlando, FL, USA, 2001, pp. 3540–3545.