Int. J. Appl. Math. and Comp. Sci., 1999, Vol.9, No.1, 129–148

SUBOPTIMAL NONLINEAR PREDICTIVE CONTROLLERS

FILIP DECLERCQ*, ROBIN DE KEYSER*

Predictive control based on linear models has become a mature technology in the last decade. Many successful real-time applications can be found in almost every sector of industry. Nonlinear predictive control can further increase the performance of this easy-to-understand control strategy. One of the main problems of implementing nonlinear predictive control is the computational aspect, which is of most importance in real-life applications. In this paper, suboptimal nonlinear predictive control strategies are proposed and compared. The nonlinear predictors are built based on neural identification methods or by white modelling. The use of diophantine equations, which is a common technique to calculate the optimal contribution of the noise model, is avoided by using a more natural method. The comparison between the control algorithms is made based on a simulated discrete multivariable nonlinear system and a continuous stirred tank reactor.

Keywords: predictive control, nonlinear control, sequential quadratic programming, diophantine equations.

1. Introduction

It is interesting to look at the history of the predictive control strategy. This strategy was not a pure theoretical approach which took years to make the switch from the academic world to the industrial world. The results presented in (Cutler and Ramaker, 1980; Richalet *et al.*, 1978) came directly from a large industrial plant, where even the slightest optimization of the plant's control strategy may result in huge profits. Although the theoretical framework around the predictive control strategy was rather limited at that time, the process industry was very interested in this easy-to-comprehend control principle. From an industrial viewpoint, the possibility to include the constraints imposed on the inputs, states and outputs directly into the formulation of the control algorithm, is an interesting property of the predictive control strategy. The predictor models which were used in the early stages were mainly linear models (impulse, step, ARMAX). Strategies using linear models are referred to in this text as linear model-based predictive control (LMBPC). The term linear only refers to the used predictor models and not to the control strategies because

* Department of Control Engineering and Automation, University Gent, Technologiepark 9, 9052 Gent, Belgium, e-mail: Filip.Declercq@rug.ac.be.

constraints can make the control action nonlinear. For more details about LMBPC the authors refer to (Camacho and Bordons, 1995; De Keyser, 1991; 1998; Garcia *et al.*, 1989).

A straightforward extension of the linear predictive control algorithm is the nonlinear model-based predictive control algorithm (NLMBPC). This strategy has first been studied thoroughly by many academic research groups before it was introduced in industrial applications (a solar plant (Camacho and Berenguel, 1994), a binary distillation column (Keeler *et al.*, 1996)). The main problem of using NLMBPC is not only the absence of a theoretical framework for an analysis but also the real-time problems which occur due to the non-convexity of the optimization problem. New techniques for modelling nonlinear processes based on neural and fuzzy modelling also opened new possibilities for NLMBPC. In this paper, a 'real' process is simulated using the so-called white model. In the first example a predictor is built using neural identification methods. Although the neural identification methods are not mature yet and there is a need for a global theoretical framework, they have already proven to be successful. It has to be said that although neural identification is based on a black box modelling approach, nonlinear system identification is more than training a neural net (Norgaard, 1995; Sjöberg *et al.*, 1995). In the second example a white model was used as predictor.

LMBPC with linear constraints can be easily tackled with quadratic programming optimization techniques (assuming that the problem is feasible). These algorithms only need a limited time to calculate the optimal feasible solution. A good overview of these optimization algorithms is given in (Wismer and Chattergy, 1978). A topic which was initially neglected during the development of NLMBPC and which needed to be solved, is the real-time aspect of the algorithm. This is necessary for a successful translation from the academic to the industrial world. Because of the nonlinear model, the strategy is described by a non-convex constrained optimization algorithm. It is obvious that NLMBPC needs much more computer power in comparison with LMBPC. It is well-known that the present optimization algorithms which can deal with non-convex problems never guarantee that the solution found is a global optimum. In (Ohtsuka and Fujii, 1997) the predictive control problem (without inequality constraints) is converted into an initial-value problem which is solved without an iterative numerical method.

In this paper, a number of algorithms is studied which are based on a simplification of the nonlinear predictor. In LMBPC the predictions $\hat{y}(t+k/t)$ of the process output $y(t+k)$ based on information known at time t and $k>0$ can be split into two parts. In the GPC framework these are usually called the free $\hat{y}_{\text{free}}(t+k/t)$ and the forced $\hat{y}_{\text{forced}}(t+k/t)$ response (Clarke *et al.*, 1987):

$$\hat{y}(t+k/t) = \hat{y}_{\text{free}}(t+k/t) + \hat{y}_{\text{forced}}(t+k/t), \quad \forall k>0 \tag{1}$$

The first term refers to the response of the predictor when the control signal $u(t+k-1) \quad \forall k>0$ is fixed to its value at $t-1$. The second term is the part of the system response which depends (linearly) on the difference of the future control signal with the control signal at time $t-1$. Because of this last property and using a quadratic cost, the optimal future control signal can be easily calculated e.g.

by using quadratic programming. In the case of a nonlinear predictor it is not possible to present the system response as a sum of the free and forced response. To avoid this problem, a number of authors have modified the NLMBPC strategy by using a simplified predictor. In (Camacho and Berenguel, 1994) a nonlinear model is used to calculate the free response of the model and one fixed incremental linearized model is used to calculate the forced response. This linear model is built around a certain state of the process. In this paper, this strategy is referred to as nonlinear generalized predictive control (NGPC) as in (Camacho and Berenguel, 1994). Another strategy that uses a similar approach is described in (Semino *et al.*, 1997) and is called nonlinear quadratic dynamic matrix control (NLQDMC). The forced response is calculated by using a linear model which is a linearization of the system around the state at time t. In this paper another approach is used. Instead of linearizing around one state, the system is linearized around a trajectory (the predicted free response) of the process. This control strategy is referred to in this paper as pseudo-NLMBPC. The pseudo-NLMBPC strategy mentioned above can be looked at as the first iteration performed for solving the nonlinear constrained optimization problem using gradient-based minimization techniques. Suboptimal-NLMBPC is a control strategy that results from a finite number of iterations of the gradient-based optimization technique. More iterations means that the suboptimal-NLMBPC algorithm results in lower values of the cost function. When the number of iterations is increased, the suboptimal-NLMBPC should approximate a local optimal solution of the non-convex constrained optimization problem. In all these algorithms a suboptimal solution is found in a finite time. This is a necessity for real-time applications. In this paper, a Levenberg-Marquardt based algorithm which can deal with constraints is used.

2. The Constrained NLMBPC Strategy

2.1. The Nonlinear Predictor

The core of the strategy is the nonlinear model of the process to be controlled. Because the paper does not emphasize the nonlinear identification problem, a general form of the nonlinear model is assumed. In this paper, the following k-step ahead predictor is presented in a state-space form but also an input-output formulation can be used:

$$\left.\begin{aligned} \hat{x}(t+k) &= f\big(\hat{x}(t+k-1), u(t+k-1)\big) \\ y(t+k) &= g\big(\hat{x}(t+k)\big) + \underbrace{\frac{C(q^{-1})}{D(q^{-1})}\varepsilon(t+k)}_{n(t+k)} \end{aligned}\right\} \forall t \geq 0 \tag{2}$$

The input vector $u(t)$ has the dimension $[nu \times 1]$, $\hat{x}(t)$ is the state vector of the predictor ($[nx \times 1]$) and $y(t)$ is the system output vector ($[ny \times 1]$). The prediction model consists of a nonlinear model $f(\cdot)$, a function $g(\cdot)$ and a noise model $\frac{C(q^{-1})}{D(q^{-1})}$, with $C(q^{-1})$ and $D(q^{-1})$ monic (matrix) polynomials. The notation $\frac{C(q^{-1})}{D(q^{-1})}$ is used instead of the more common notation $D(q^{-1})^{-1}C(q^{-1})$ for representing a multi-input

multi-output system. The prediction error $\varepsilon(t) = y(t) - \hat{y}(t/t-1)$ is assumed to be a vector of white noise with average value 0 and $\hat{y}(t/t-1)$ is the one-step ahead prediction of $y(t)$ at time $t-1$. This model structure is closely related to a Box-Jenkins model (when $f(\cdot)$ and $g(\cdot)$ are linear functions) which is well-known in the world of linear system identification. This model can be used when the outputs of the nonlinear system are corrupted by colored noise which is assumed to be generated by a linear noise model $\frac{C(q^{-1})}{D(q^{-1})}$. It is clear that (2) cannot be used directly to calculate the value of $y(t+k)$ because the value of the colored noise $n(t+k)$ is unknown at time t. Deriving the optimal predictor in the MBPC algorithm is closely related to the derivation of the optimal one-step ahead estimator in prediction error identification algorithms (Ljung, 1987). In (Clarke *et al.*, 1987) an ARMAX model was used and in order to derive an analytical solution of GPC the usage of diophantine equations was introduced. To make a k-step-ahead optimal estimation of $n(t+k)$ the noise model is split into two parts using a diophantine equation

$$n(t+k) = E_k(q^{-1})\varepsilon(t+k) + \underbrace{q^{-k}\frac{F_k(q^{-1})}{D(q^{-1})}\varepsilon(t+k)}_{n(t+k/t)} \tag{3}$$

The first part contains the information unknown at time t; the second part depends only on data known at time t. $E_k(q^{-1})$ is a (matrix) polynomial of degree $k-1$.

The optimal prediction or the conditional expectation (Ljung, 1987) of $y(t+k)$ is defined by

$$\left.\begin{aligned} \hat{x}(t+k) &= f\big(\hat{x}(t+k-1), u(t+k-1)\big) \\ \hat{y}(t+k/t) &= g\big(\hat{x}(t+k)\big) + n(t+k/t) \end{aligned}\right\} \forall t \geq 0 \tag{4}$$

This is only one method to calculate the optimal prediction $\hat{y}(t+k/t)$. Another straightforward method which is used in this paper is by directly filtering the optimal estimation of the prediction error $\varepsilon(t+k/t)$ $\forall t, k$. The optimal estimation of the k-step-ahead prediction error $\varepsilon(t+k/t)$ based on the information available at time t is defined by

$$\left.\begin{aligned} \varepsilon(t+k/t) &= \varepsilon(t+k) = y(t+k) - \hat{y}(t+k/t+k-1) \quad \forall k \leq 0 \\ \varepsilon(t+k/t) &= E[\varepsilon(t+k)] = 0 \quad \forall k > 0 \end{aligned}\right\} \tag{5}$$

where $E[\cdot]$ stands for the expectation. From the theory of statistics it is known that the optimal prediction of a white stochastic process is its mean value (Lewis, 1986).

Lemma 1. *The k-step-ahead optimal prediction $n(t+k/t)$ of the noise $n(t+k)$ at time instant t is the response of the noise model $\frac{C(q^{-1})}{D(q^{-1})}$ which is driven by the optimal prediction error $\varepsilon(t+k/t)$ and initialized by the colored noise $n(t+k)$ $(k \leq 0)$ which is known at time instant t,*

$$n(t+k/t) = \frac{C(q^{-1})}{D(q^{-1})}\varepsilon(t+k/t) \quad \forall k, t \tag{6}$$

The corresponding proof is given in the Appendix. Lemma 1 is now used to calculate the optimal prediction of the output in a more natural way.

2.2. The Mathematical Description of the NLMBPC Strategy

To write the cost function of the NLMBPC strategy in a compact manner, some vectors are to be defined. The future state output of the nonlinear predictor is defined by

$$\bar{\mathbf{x}} = \underbrace{\left[\hat{x}_1(t+L_0), \ldots, \hat{x}_{nx}(t+L_0), \hat{x}_1(t+L_0+1), \ldots, \hat{x}_1(t+L_y), \ldots, \hat{x}_{nx}(t+L_y)\right]^T}_{[nx\cdot(L_y-L_0+1)\times 1]} \quad (7)$$

L_0 and L_y being the minimum and maximum prediction horizon, respectively. The future outputs $\bar{\mathbf{y}}$ and future setpoints $\bar{\mathbf{w}}$ have a similar structure as $\bar{\mathbf{x}}$. The future control sequence is defined by

$$\bar{\mathbf{u}} = \underbrace{\left[u_1(t), \ldots, u_{nu}(t), u_1(t+1), \ldots, u_1(t+L_u-1), \ldots, u_{nu}(t+L_u-1)\right]^T}_{[nu\cdot L_u\times 1]} \quad (8)$$

The control horizon L_u is used to limit the number of future control moves. This means that

$$\Delta u_j(t+k) = 0 \quad \text{for } j = 1, \ldots, nu \quad \text{and } L_u \leq k < L_y \quad (9)$$

where $\Delta = 1 - q^{-1}$ and q^{-1} is the backward shift operator. The quality of the control action $\bar{\mathbf{u}}$ (at time t) is measured using the cost function

$$J(\bar{\mathbf{u}}) = \frac{1}{2}\bar{\mathbf{e}}^T\bar{\mathbf{e}} + \frac{1}{2}\rho\Delta\bar{\mathbf{u}}^T\Delta\bar{\mathbf{u}} \quad (10)$$

ρ being the weight for the control sequence. The error vector $\bar{\mathbf{e}}$ has the same nature as $\bar{\mathbf{y}}$ and is defined by $\bar{\mathbf{e}} = P(q^{-1})\bar{\mathbf{y}} - R(q^{-1})\bar{\mathbf{w}}$. The transfer functions in the cost function are defined by $P(q^{-1}) = 1 - \alpha q^{-1}$ and $R(q^{-1}) = 1 - \alpha$ were α acts as the discrete pole of the reference system.

The constraints imposed on the control problem (e.g. $y(t) \leq y_{\max}$) are represented in a general form by

$$\left.\begin{aligned} g_1(\bar{\mathbf{u}}, \bar{\mathbf{y}}, \bar{\mathbf{x}}) &= 0 \\ g_2(\bar{\mathbf{u}}, \bar{\mathbf{y}}, \bar{\mathbf{x}}) &\leq 0 \end{aligned}\right\} \quad (11)$$

At time t the optimal control vector is the vector $\bar{\mathbf{u}}$ which minimizes the cost function (10) taking into account the constraints (11). Then the first nu elements of the optimal control vector $\bar{\mathbf{u}}$ are applied to the process and the whole optimization procedure is repeated at $t+1$.

2.3. A General Framework for Deriving a Suboptimal Solution of the Cost (10) with the Constraints (11)

Minimization of (10) with respect to $\bar{\mathbf{u}}$ is performed by using a gradient-based method. The core of such an iterative optimization algorithm is the calculation of an incremental change $\delta\bar{\mathbf{u}}_i$ (during the $(i+1)$-th iteration) so that $J(\bar{\mathbf{u}}_i+\delta\bar{\mathbf{u}}_i) < J(\bar{\mathbf{u}}_i)$ with $\bar{\mathbf{u}}_i$ being the future control vector of the previous (i-th iteration). The Levenberg-Marquardt optimization method uses implicitly a model of the nonlinear predictor which is linear in $\delta\bar{\mathbf{u}}_i$ (Declercq and De Keyser, 1995). A 'general' form of such a linearized predictor is as follows:

$$\bar{\mathbf{y}}(\bar{\mathbf{u}}_i+\delta\bar{\mathbf{u}}_i) \approx \bar{\mathbf{y}}(\bar{\mathbf{u}}_i) + \left(\frac{\partial\bar{\mathbf{y}}}{\partial\bar{\mathbf{x}}}\frac{\partial\bar{\mathbf{x}}}{\partial\bar{\mathbf{u}}}\bigg|_{\bar{\mathbf{u}}^*,\bar{\mathbf{x}}^*}\right)^T \delta\bar{\mathbf{u}}_i \tag{12}$$

The term 'general' refers to the possibility of easily transforming (12) into different simplified predictors. This is done by making a specific choice of the vectors $\bar{\mathbf{x}}^*$ and $\bar{\mathbf{u}}^*$ (see Section 2.4).

From eqn. (12) it is clear that the response of the predictor due to $\delta\bar{\mathbf{u}}_i$ is linearized along $\bar{\mathbf{u}}^*$ and $\bar{\mathbf{x}}^*$. Note that the right part of eqn. (12) becomes the first-order Taylor expansion of $\bar{\mathbf{y}}(\bar{\mathbf{u}}_i+\delta\bar{\mathbf{u}}_i)$ when $\bar{\mathbf{u}}^*$ is replaced by $\bar{\mathbf{u}}_i$ and $\bar{\mathbf{x}}^*$ is replaced by $\bar{\mathbf{x}}_i$.

Inserting (12) in (10) and adding (as proposed by Levenberg) a quadratic cost of the form $\frac{1}{2}\mu\delta\bar{\mathbf{u}}_i^T\delta\bar{\mathbf{u}}_i$ to the considered cost result in

$$J_{\text{lin}}(\delta\bar{\mathbf{u}}_i) = \frac{1}{2}\delta\bar{\mathbf{u}}_i^T(G+\mu\mathbf{I})\delta\bar{\mathbf{u}}_i + g^T\delta\bar{\mathbf{u}}_i \tag{13}$$

with G and g defined by

$$G = \left(\left[\frac{\partial\bar{\mathbf{e}}^{\mathbf{T}}}{\partial\bar{\mathbf{u}}}\bigg|_{\bar{\mathbf{u}}^*,\bar{\mathbf{x}}^*} \quad \sqrt{\rho}\frac{\partial\Delta\bar{\mathbf{u}}^T}{\partial\bar{\mathbf{u}}}\right]\left[\frac{\partial\bar{\mathbf{e}}^{\mathbf{T}}}{\partial\bar{\mathbf{u}}}\bigg|_{\bar{\mathbf{u}}^*,\bar{\mathbf{x}}^*} \quad \sqrt{\rho}\frac{\partial\Delta\bar{\mathbf{u}}^T}{\partial\bar{\mathbf{u}}}\right]^T\right) \tag{14}$$

$$g = \frac{\partial\bar{\mathbf{e}}^{\mathbf{T}}}{\partial\bar{\mathbf{u}}}\bigg|_{\bar{\mathbf{u}}^*,\bar{\mathbf{x}}^*}\bar{\mathbf{e}}|_{\bar{\mathbf{u}}_i,\bar{\mathbf{x}}_i} + \rho\,\frac{\partial\Delta\bar{\mathbf{u}}^T}{\partial\bar{\mathbf{u}}}\Delta\bar{\mathbf{u}}\bigg|_{\bar{\mathbf{u}}_i,\bar{\mathbf{x}}_i} \tag{15}$$

In an (unconstrained) Levenberg-Marquardt optimization algorithm, the incremental change $\delta\bar{\mathbf{u}}_i$ in the control vector is the solution to the quadratic cost (13).

The Levenberg-Marquardt optimization algorithm which is normally used for unconstrained optimization can be extended so that it can also be used for constrained optimization problems. When there are constraints, $\delta\bar{\mathbf{u}}_i$ is calculated by solving the quadratic programming problem defined by

$$\left.\begin{aligned}
&\min_{\delta\bar{\mathbf{u}}_i} J_{\text{lin}}(\delta\bar{\mathbf{u}}_i) = \frac{1}{2}\delta\bar{\mathbf{u}}_i^T(G+\mu I)\delta\bar{\mathbf{u}}_i + g^T\delta\bar{\mathbf{u}}_i \\
&g_1(\bar{\mathbf{u}},\bar{\mathbf{y}},\bar{\mathbf{x}})|_{\bar{\mathbf{u}}_i,\bar{\mathbf{x}}_i} + \left(\frac{\partial g_1}{\partial\bar{\mathbf{u}}} + \frac{\partial g_1}{\partial\bar{\mathbf{x}}}\frac{\partial\bar{\mathbf{x}}}{\partial\bar{\mathbf{u}}} + \frac{\partial g_1}{\partial\bar{\mathbf{y}}}\frac{\partial\bar{\mathbf{y}}}{\partial\bar{\mathbf{u}}}\right)\bigg|_{\bar{\mathbf{u}}^*,\bar{\mathbf{x}}^*}\delta\bar{\mathbf{u}}_i = 0 \\
&g_2(\bar{\mathbf{u}},\bar{\mathbf{y}},\bar{\mathbf{x}})|_{\bar{\mathbf{u}}_i,\bar{\mathbf{x}}_i} + \left(\frac{\partial g_2}{\partial\bar{\mathbf{u}}} + \frac{\partial g_2}{\partial\bar{\mathbf{x}}}\frac{\partial\bar{\mathbf{x}}}{\partial\bar{\mathbf{u}}} + \frac{\partial g_2}{\partial\bar{\mathbf{y}}}\frac{\partial\bar{\mathbf{y}}}{\partial\bar{\mathbf{u}}}\right)\bigg|_{\bar{\mathbf{u}}^*,\bar{\mathbf{x}}^*}\delta\bar{\mathbf{u}}_i \le 0
\end{aligned}\right\} \tag{16}$$

The quadratic programming problem (16) is solved using the `qp.m` procedure of (Grace, 1995). The constraints in (16) are obtained using the same methodology as for the 'general' predictor (12). The constraints (11) are linearized but the part related to the future control moves is defined by the vectors $\bar{\mathbf{u}}^*$ and $\bar{\mathbf{x}}^*$. The new control sequence $\bar{\mathbf{u}}_{i+1}$ is then calculated by

$$\bar{\mathbf{u}}_{i+1} = \bar{\mathbf{u}}_i + \delta\bar{\mathbf{u}}_i \tag{17}$$

If $J(\bar{\mathbf{u}}_{i+1}) < J(\bar{\mathbf{u}}_i)$, the value of $\bar{\mathbf{u}}_{i+1}$ is accepted and μ is decreased. If $J(\bar{\mathbf{u}}_{i+1}) \geq J(\bar{\mathbf{u}}_i)$, then the quadratic programming problem (16) is solved again with an increased value for the parameter μ. The basic idea for updating the μ parameter is based on the fact that increasing μ moves $\delta\bar{\mathbf{u}}_i$ towards the direction of the steepest descent and reduces the step length. Decreasing μ results in a $\delta\bar{\mathbf{u}}_i$ that swings towards the Gauss-Newton direction. The way of updating the value of μ is described in the flowchart of the algorithm (Fig. 1) and is based on the technique used in (Demuth and Beale, 1994). The optimization algorithm is initialized ($i = 0$) using the control vector $\bar{\mathbf{u}}_0$ which contains only the control action $u(t-1)$ known at time t. This procedure is repeated until a minimum is found or when the maximum number of iterations ($\text{epoch}_{\max}$) has been exceeded. The optimization algorithm is also stopped when the norm of the gradient g is smaller than a predefined value $g_{\min}$ or when the value of the cost $J(\bar{\mathbf{u}}_i)$ is smaller than a predefined value $J_{\min}$ or when μ has reached its maximum value $\mu_{\max}$.

2.4. The Vectors $\bar{\mathbf{u}}^*$ and $\bar{\mathbf{x}}^*$

By using the flowchart of Fig. 1 and by making a specific choice of the vectors $\bar{\mathbf{u}}^*$ and $\bar{\mathbf{x}}^*$ the different algorithms mentioned in the Introduction can be generated.

2.4.1. NGPC and NLQDMC

The algorithm NGPC presented in (Camacho and Berenguel, 1994) can be generated when in eqn. (16) the vectors $\bar{\mathbf{u}}^*$ and $\bar{\mathbf{x}}^*$ are fixed and when they contain only the control signals and the state of the process around which the incremental model was initially built. This means that the forced response is a linear function in $\delta\bar{\mathbf{u}}_0$. Because of the initialization of the control vector with $\bar{\mathbf{u}}_0$, the first vector of the right part of eqn. (12) is the free response of the nonlinear predictor.

The NLQDMC strategy can be generated when the vector $\bar{\mathbf{x}}^*$ contains only the state of the process at time t and $\bar{\mathbf{u}}^* = \bar{\mathbf{u}}_0$. As in NGPC, the forced response is a linear function in $\delta\bar{\mathbf{u}}_0$ but it is built up based on the model linearized at the current state of the process. Using the general framework (Subsection 2.3), only one iteration (i.e. $\text{epoch}_{\max} = 1$) of the constrained L-M optimization algorithm is needed to generate NGPC and NLQDMC, i.e. the value $\bar{\mathbf{u}}_0 + \delta\bar{\mathbf{u}}_0$ calculated after one iteration is applied to the process. The same idea is used in the pseudo-NLMBPC algorithm.

2.4.2. Pseudo-NLMBPC

In this algorithm $\bar{\mathbf{x}}^*$ is the response of the predictor to the input $\bar{\mathbf{u}}^* = \bar{\mathbf{u}}_0$. This means that $\bar{\mathbf{x}}^*$ represents the free response (of the states) of the predictor and that the

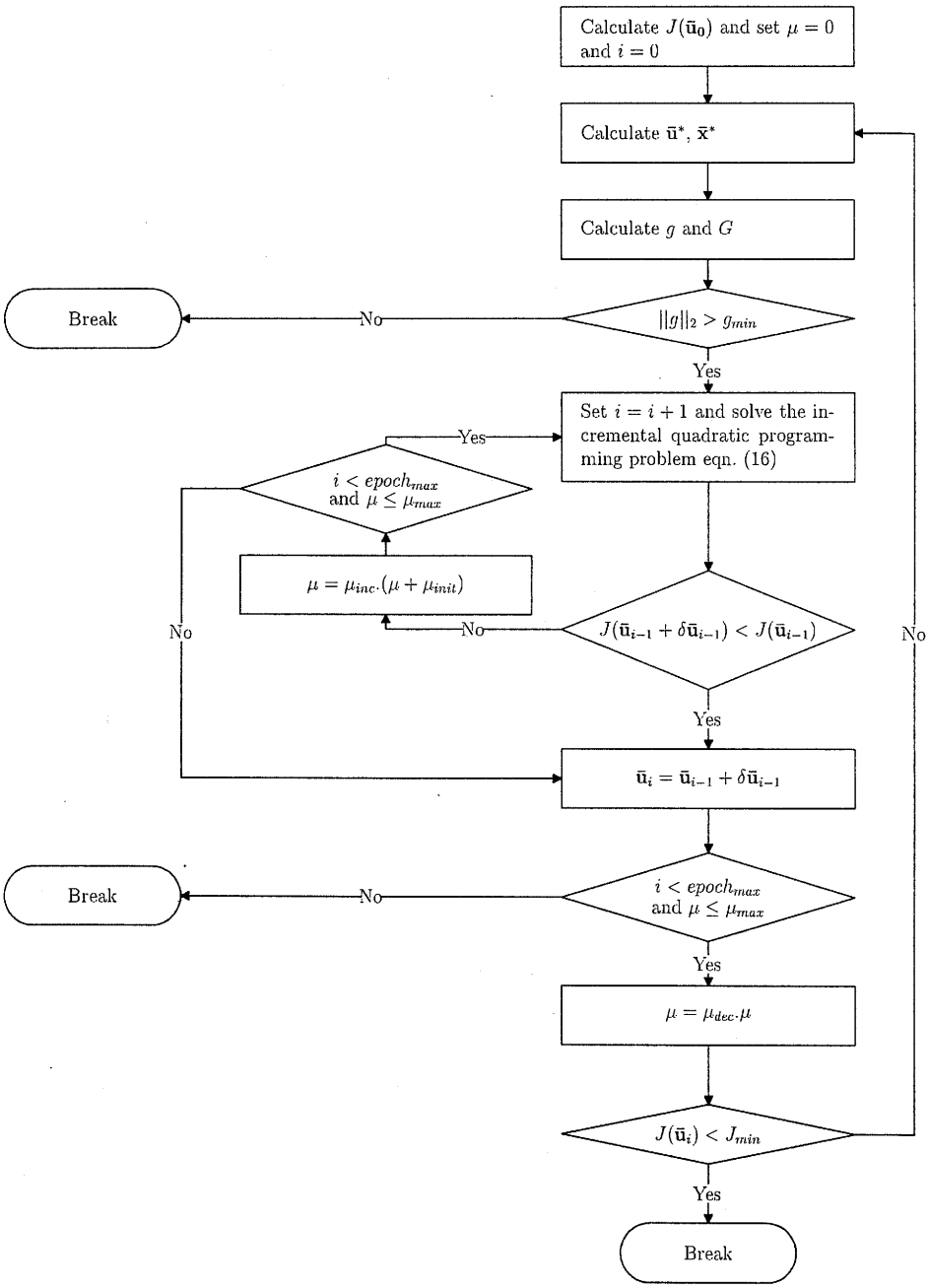

Fig. 1. Optimization algorithm.

process is linearized around this free response. Since the cost using pseudo-NLMBPC is lower than when NGPC or NLQDMC are used, this results in a better control strategy, especially when the system is highly nonlinear around the trajectory of the free response. The differences between NGPC, NLQDMC and pseudo-NLMBPC can be clearly seen in the partial derivatives matrix $\frac{\partial \bar{\mathbf{y}}}{\partial \bar{\mathbf{x}}} \frac{\partial \bar{\mathbf{x}}}{\partial \bar{\mathbf{u}}}|_{\bar{\mathbf{u}}^*, \bar{\mathbf{x}}^*}$ (eqn. (18) when $L_0 = 1$). This matrix is built up with the (matrix) impulse response coefficients h_l^k ($[ny \times nu]$) being the l-th impulse response coefficients of the linearized model at the k-th prediction step. Note that these linearized models are not calculated explicitly during the algorithm. In NGPC $\frac{\partial \bar{\mathbf{y}}}{\partial \bar{\mathbf{x}}} \frac{\partial \bar{\mathbf{x}}}{\partial \bar{\mathbf{u}}}|_{\bar{\mathbf{u}}^*, \bar{\mathbf{x}}^*}$ is fixed during the control (independent of the time) because $\bar{\mathbf{u}}^*$ and $\bar{\mathbf{x}}^*$ do not change during the control. In pseudo-NLMBPC and NLQDMC the impulse response coefficients are recalculated every sampling period, because the values of $\bar{\mathbf{u}}^*$ and $\bar{\mathbf{x}}^*$ change. When NLQDMC is used, the impulse response coefficients

$$\left.\frac{\partial \bar{\mathbf{y}}}{\partial \bar{\mathbf{x}}} \cdot \frac{\partial \bar{\mathbf{x}}}{\partial \bar{\mathbf{u}}}\right|_{\bar{\mathbf{u}}^*, \bar{\mathbf{x}}^*} = \begin{bmatrix} h_1^1 & 0 & \dots & 0 \\ h_2^2 & h_1^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ h_{Lu}^{Lu} & h_{Lu-1}^{Lu} & \dots & h_1^{Lu} \\ h_{Lu+1}^{Lu+1} & h_{Lu}^{Lu+1} & \dots & h_1^{Lu+1} + h_2^{Lu+1} \\ \vdots & \vdots & \vdots & \vdots \\ h_{Ly}^{Ly} & h_{Ly-1}^{Ly} & \dots & \sum_{k=1}^{Ly-L_u+1} h_k^{Ly} \end{bmatrix} \tag{18}$$

fulfil the simple relationship $h_l^k = h_l^j \ \forall l$.

2.4.3. Suboptimal-NLMBPC

When a process is highly nonlinear over the prediction horizon, it is possible that the results of the pseudo-NLMBPC are poor. The performance of the control action can be increased by using a control vector $\bar{\mathbf{u}}_i$ such that the cost $J(\bar{\mathbf{u}}_i)$ is sufficiently reduced. This is done by applying only a finite number ($\text{epoch}_{\max}$) of searches when minimizing the constrained optimization algorithm. This number is called the level of the suboptimal control strategy. The vector $\bar{\mathbf{x}}^*$ changes during the optimization: during the first iteration $\bar{\mathbf{x}}^*$ contains the free response of the predictor, during the i-th iteration $\bar{\mathbf{x}}^*$ is the response of the predictor when the control vector $\bar{\mathbf{u}}_{i-1}$ (derived during the $(i-1)$-th iteration) is applied to the predictor.

2.4.4. NLMBPC

In this paper, NLMBPC is the control action which minimizes the constrained cost function. Due to the already mentioned optimization problems, the NLMBPC is

approximated by a suboptimal-NLMBPC of a sufficiently high level, e.g. $\text{epoch}_{\max} = 200$. Note that due to the non-convex nature of the problem it is not sure that the minimum found by this algorithm is global.

3. Illustrative Examples

3.1. Narendra's 5-th System

The first example is a MIMO system which was used in (Narendra and Parthasarathy, 1990) and is often used as a benchmark for neural identification methods. The system has no real physical meaning but it is highly nonlinear and coupled. The system is described by two difference equations:

$$\begin{aligned} x_1(k+1) &= \frac{x_1(k)}{1+x_2(k)^2} + u_1(k) \\ x_2(k+1) &= \frac{x_1(k)x_2(k)}{1+x_2(k)^2} + u_2(k) \end{aligned} \tag{19}$$

In this example a neural model was used in the nonlinear predictor to estimate $f(\cdot)$. In (Declercq and De Keyser, 1997) the neural identification method used is described. The diagonal elements of the polynomial matrices of the noise model are set as $C_{ii}(q) = 1$ and $D_{ii}(q) = 1 - q^{-1}$ $(i = 1, 2)$. Two constraints are imposed on the system, i.e.

$$y_1(t) \leq 0.8, \quad y_2(t) \leq 0.5 \quad \forall t \tag{20}$$

The parameters of the cost (10) are set as follows: $L_y = 10$, $L_o = 1$, $L_u = 3$, $\alpha = 0.5$ and $\rho = 0$. The parameters for updating μ in the optimization algorithm are set to $\mu_{\text{init}} = 1 \times 10^{-3}$, $\mu_{\text{inc}} = 10$, $\mu_{\text{dec}} = 1 \times 10^{-1}$. The parameters $g_{\min}$, $J_{\min}$ and $\mu_{\max}$ which determine when a minimum of the NLMBPC cost is found, are set to $\mu_{\max} = 1 \times 10^{10}$, $g_{\min} = 1 \times 10^{-10}$, $J_{\min} = 1 \times 10^{-10}$. These values were selected based on simulation experiments.

In Fig. (2) the control results are plotted for the pseudo-NLMBPC, suboptimal-NLMBPC and NLMBPC. The plot in the upper-left panel shows the results of the pseudo-NLMBPC strategy of Subsection 2.4.2 (linearized around the free response). The control results are good but there is a ripple at time 17 in $x_2(t)$ and the decoupling at time 55 is not completely satisfied. If a suboptimal-NLMBPC strategy of the second level (upper-right panel) is applied, the ripple at time 17 is removed and the decoupling is improved. It is clear that the difference between the suboptimal-NLMBPC of the third level (lower-left panel) and NLMBPC (level 200) (lower-right panel) is small. This means that only a few iterations are needed to find a good suboptimal solution of the control vector. In Fig. (3) the results of the control used in (Camacho and Berenguel, 1994) are presented (NGPC). The incremental model was built around the origin. In Fig. (4) the linear incremental model was built around the current state of the system at every sampling instant t (NLQDMC). It is clear that the results of these control strategies (that use a linear model to predict the forced response) are not as good as when the process is linearized around the free response.

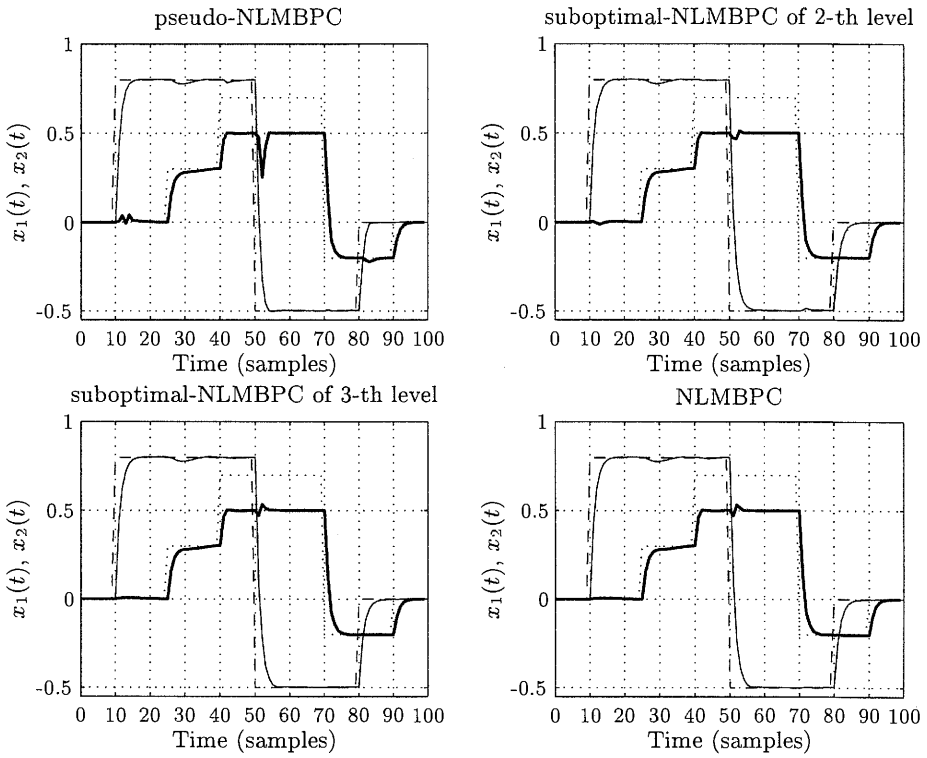

Fig. 2. Control results using pseudo-NLMBPC, suboptimal-NLMBPC and NLMBPC strategy, with $x_1(t)$ (thin solid line) and $x_2(t)$ (thick solid line).

Note that the approximations used to simplify the predictor are also present in the constraints of the quadratic programming problem (16). The performance of NGPC and NLQDMC degrades when the difference between the linearized models along the trajectory of the free response becomes higher. It is interesting to note that in the case of a linear predictor all strategies are identical.

3.2. Control of a Continuous Stirred Tank Reactor (CSTR)

The dynamics of the process are represented by four nonlinear differential equations (Chen *et al.*, 1995; Engell and Klatt, 1993):

$$\dot{x}_1 = u_1(t)\big[x_{10}(t) - x_1(t)\big] - k_1\big(x_3(t)\big)x_1(t) - k_3\big(x_3(t)\big)x_1(t)^2 \tag{21}$$

$$\dot{x}_2 = -u_1(t)x_2(t) + k_1\big(x_3(t)\big)x_1(t) - k_2\big(x_3(t)\big)x_2(t) \tag{22}$$

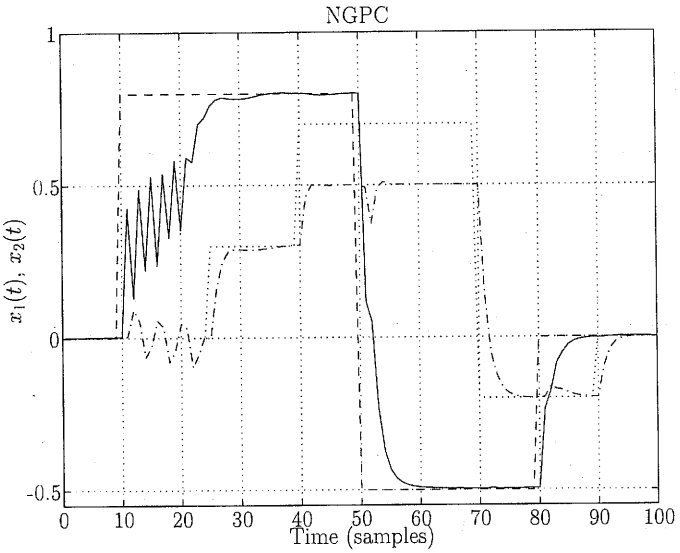

Fig. 3. NGPC $x_1(t)$ (solid line) and $x_2(t)$ (dash-dot line).

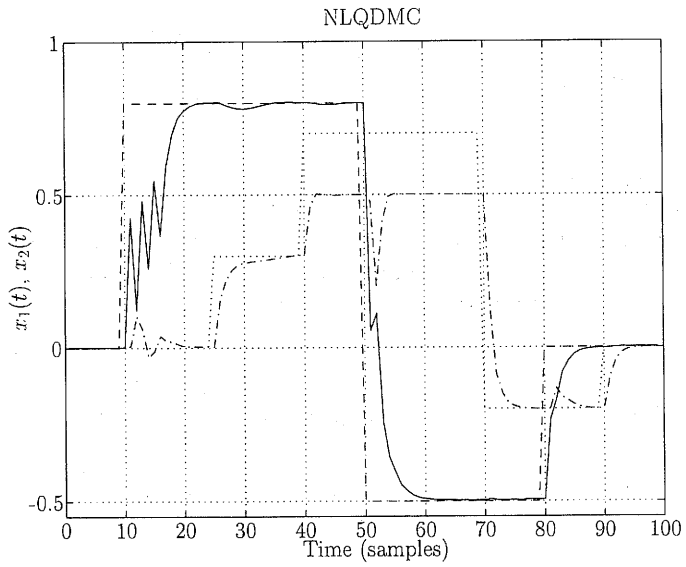

Fig. 4. NLQDMC $x_1(t)$ (solid line) and $x_2(t)$ (dash-dot line).

$$\dot{x}_3 = u_1(t)\big[x_{30}(t) - x_3(t)\big] + \frac{k_w A_R}{\rho C_p V_R}\big[x_4(t) - x_3(t)\big]$$
$$- \frac{1}{\rho C_p}\Big[\Delta H_{R_{AB}} k_1\big(x_3(t)\big)x_1(t)$$
$$+ \Delta H_{R_{BC}} k_2\big(x_3(t)\big)x_2(t) + \Delta H_{R_{AD}} k_3\big(x_3(t)\big)x_1(t)^2\Big] \tag{23}$$

$$\dot{x}_4 = \frac{1}{m_K C_{PK}}\Big[u_2(t) + k_w A_R\big(x_3(t) - x_4(t)\big)\Big] \tag{24}$$

with $k_i(x_3(t))$, $i = 1,2,3$ coming from the Arrhenius law

$$k_i(x_3(t)) = k_{i0} \exp\left(\frac{E_i}{273.15 + x_3(t)}\right), \quad i = 1,2,3 \tag{25}$$

and the other model parameters as listed in (Chen *et al.*, 1995).

The process model has four inputs (i.e. the normalized feed flow $u_1(t)$ (h^{-1}), the heat removal $u_2(t)$ (kJ/h), the feed temperature $x_{30}(t)$ (°) and the concentration of the initial reactant $x_{10}(t)$ (mol/l)). In this paper, the inputs $x_{30}(t)$ and $x_{10}(t)$ are assumed constant during the experiment. The process has four states $x(t) = [x_1(t)\ x_2(t)\ x_3(t)\ x_4(t)]^T$. $x_1(t)$ is the concentration in mol/l of the educt and $x_2(t)$ is the concentration in mol/l of the desired product. The temperature of the contents of the reactor is $x_3(t)$ in (°) and $x_4(t)$ is the temperature of the reactor coolant (°). The task of the controller is to produce an output flow which has a certain concentration $x_2(t)$ and temperature $x_3(t)$ in the reactor. The steady state point of the process is $x_0 = [2.14\ 1.09\ 114.2\ 112.9]^T$ when the inputs are set to $u_1(0) = 14.19$, $u_2(0) = -1113.5$, $x_{30}(0) = 104.9$ and $x_{10}(0) = 5.10$. This point is also referred to as the point of maximum yield in (Chen *et al.*, 1995). In Figs. 5 and 6 the steady state behavior of the CSTR is depicted and the point of the maximum yield is marked by the top of the thick line. A zero-order-hold with a sampling period of 20 s is placed in front of the continuous system. The differential equations are integrated using a fifth-order Runge-Kutta method with fourth-order step-size control, i.e. the Matlab procedure `rk45.m`. The function $g(\cdot)$ is for this example (MIMO-control) defined by

$$y(t) = Cx(t) \tag{26}$$

where

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \end{bmatrix}$$

The range of the two control signals is limited, namely

$$\left.\begin{aligned} 5 \;&\leq u_1(t) \leq 35 \\ -9000 \;&\leq u_2(t) \leq 0 \end{aligned}\right\} \forall t \tag{27}$$

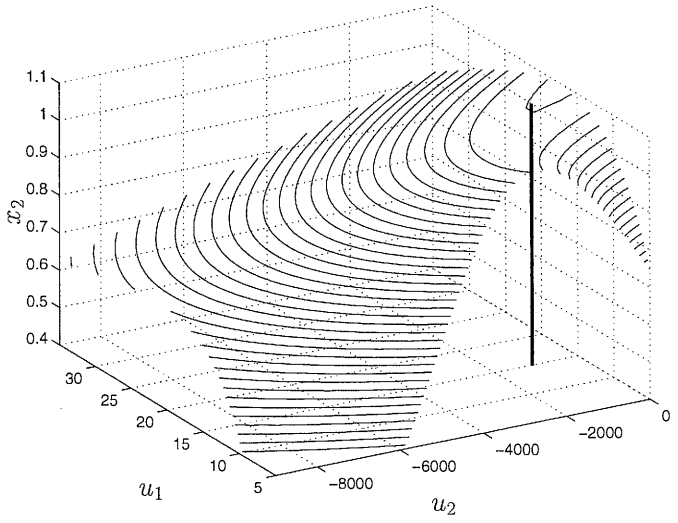

Fig. 5. Steady-state values of x_2.

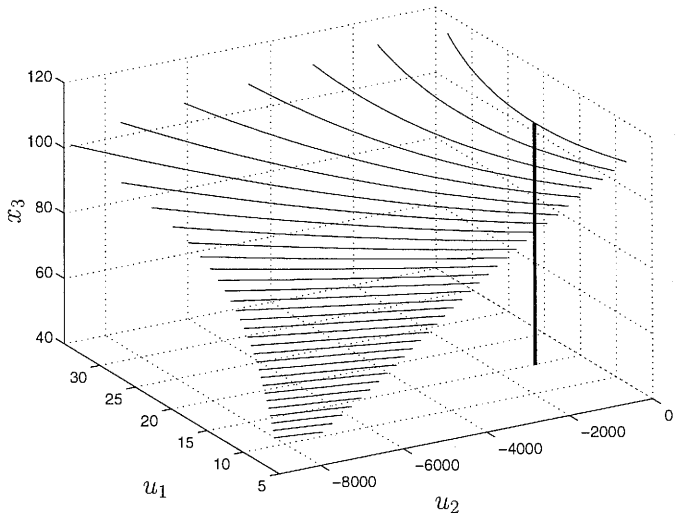

Fig. 6. Steady-state values of x_3.

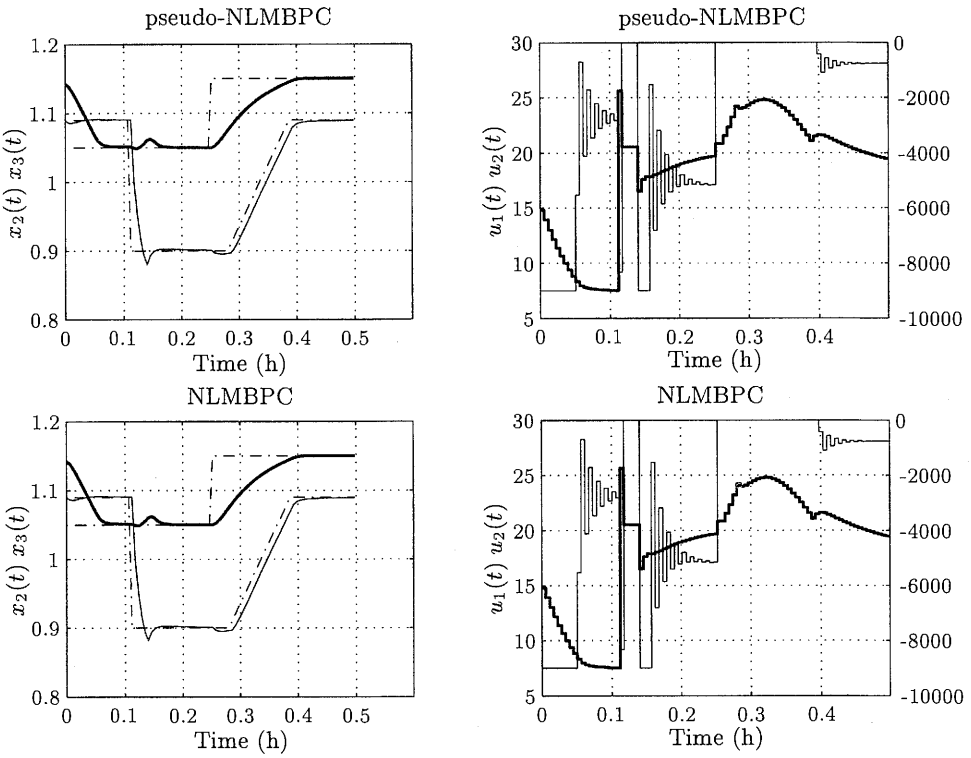

Fig. 7. Control result using pseudo-NLMBPC and (suboptimal)-NLMBPC ($\text{epoch}_{\max} = 200$) with $x_1(t)$ (thin line), $x_3(t)/100$ (thick line), $u_1(t)$ (thick line, using left scale), $u_2(t)$ (thin line, using right scale).

The control results of the CSTR using pseudo-NLMBPC and NLMBPC are depicted in Fig. 7. The parameters of the cost (10) are the same as in the previous example except for the value of the control horizon L_u which was set to 5. A white model was used in the predictor with slightly perturbated parameters when compared with those used in the 'real'-process (Chen *et al.*, 1995, Case 1). In this example the predictor is configured as an input-output model, i.e. it is not initialized with the measured state of the process but with the output of the predictor at time $t-1$. When the predictor is initialized with the state of the process $x(t)$ and there is a model mismatch between the real process and predictor, an offset will occur in the control results even when an extended Kalman filter is used to optimize the measured state $x(t)$. This was also noted in (Semino *et al.*, 1997).

From Fig. 7 it is clear that there is a small difference between the pseudo-NLMBPC and (suboptimal)-NLMBPC for this example. The reason lies probably in the fast dynamics of the system, i.e. the coefficients h_1^j with $j = 1, \ldots, L_u$ in eqn. (18) are much larger than the other coefficients. The fact that the difference

between pseudo-NLMBPC and NLMBPC is small (non-detectable by eye) makes it also possible to use large (quasi-infinite) prediction horizons as proposed in (Chen and Allgöwer, 1997) even in real-time control applications.

4. Conclusions

A number of nonlinear model based predictive control strategies are presented and compared with strategies presented by other authors. The proposed pseudo-NLMBPC strategy acts as the first iteration in a constrained Levenberg-Marquardt based optimization algorithm. In spite of its simplicity, the control results are very good. An optimization algorithm which can deal with the constrained non-linear least-squares problem of the NLMBPC was introduced in this paper. It is based on a sequential quadratic programming method and the Levenberg-Marquardt algorithm.

The performance of the pseudo-NLMBPC can be improved by using suboptimal-NLMBPC strategies. These are algorithms with a finite (often very small) number of maximum iterations in the optimization algorithm. These algorithms have a finite calculation time which makes them useful for real-time implementations. The results of these strategies lay between those of the pseudo-NLMBPC and 'exact'-NLMBPC. The simulations of two highly nonlinear MIMO systems were used to illustrate the control algorithms.

Appendix

To proof Lemma 1, the equivalence between the diophantine equations and the lemma is shown. First, based on a proof by induction a general form

$$n(t+k/t) = \left(\frac{C(q^{-1}) - D(q^{-1})E_k(q^{-1})}{D(q^{-1})} \right) \varepsilon(t+k) \tag{A1}$$

is derived which uses an iterative formula to calculate

$$\begin{aligned} E_k(q^{-1}) &= E_{k-1}(q^{-1}) + C_k(q^{-1}) \\ &\quad - \sum_{j=0}^{k-2} D_{k-j}(q^{-1}) \left[E_j(q^{-1}) - E_{j+1}(q^{-1}) \right] \end{aligned} \tag{A2}$$

and taking into account that $E_0(q^{-1}) = 0$. To complete the proof, it is shown that this general form is equivalent with the diophantine equations (3).

To make the proof more compact, the (matrix) polynomial $D(q^{-1})$ is denoted by D and D_k is defined as the polynomial which contains the first k coefficients of the original $D(q^{-1})$ polynomial:

$$D(q^{-1}) = I + d_1 q^{-1} + \cdots + d_{k-1} q^{-k+1} + \cdots + d_{nd} q^{-nd} \tag{A3}$$

$$D_k(q^{-1}) = I + d_1 q^{-1} + \cdots + d_{k-1} q^{-k+1} \tag{A4}$$

In the case of a multivariable noise model, I is the identity matrix and d_k becomes a square matrix of dimensions $[ny \times ny]$.

During the proof the following relationships are used:

$$\left.\begin{aligned}(D - D_k)n(t + k/t) &= (D - D_k)n(t + k)\\ (C - C_k)\varepsilon(t + k/t) &= (C - C_k)\varepsilon(t + k)\end{aligned}\right\} \tag{A5}$$

The notation '$/t$' can thus be omitted here due to the fact that $(D_k - D)n(t + k/t)$ and $(C - C_k)\varepsilon(t + k/t)$ contain only information known at time t.

Proof of Lemma 1. The optimal one-step ahead prediction of $n(t + 1)$ using the lemma gives

$$n(t + 1/t) = (D_1 - D)n(t + 1/t) + (C - C_1)\varepsilon(t + 1/t) + C_1\varepsilon(t + 1/t) \tag{A6}$$

In the first and second term on the right-hand side '$/t$' can be omitted (see eqn. (A5)). Using the definition of the optimal prediction error the third term is equal to zero. This results in

$$n(t + 1/t) = (D_1 - D)n(t + 1) + (C - C_1)\varepsilon(t + 1) \tag{A7}$$

Using the definition of $n(t + k)$ this can be written as

$$n(t + 1/t) = \left(\frac{C - DC_1}{D}\right)\varepsilon(t + 1) \tag{A8}$$

This gives the same result as (A1) and (A2).

Assuming that eqn. (A1) is correct for $n(t + k/t)$, we have to show that $n(t + k + 1/t)$ has a similar form. Using the lemma, we have

$$\begin{aligned}n(t + k + 1/t) &= (D_{k+1} - D)n(t + k + 1/t) - (D_{k+1} - I)n(t + k + 1/t)\\ &\quad + (C - C_{k+1})\varepsilon(t + k + 1/t) + C_{k+1}\varepsilon(t + k + 1/t)\end{aligned} \tag{A9}$$

In the first and third term on the right-hand side '$/t$' can be omitted (see eqn. (A5)). Using the definition of the optimal prediction error, the fourth term is equal to zero which results in

$$\begin{aligned}n(t + k + 1/t) &= (D_{k+1} - D)n(t + k + 1) - (D_{k+1} - I)n(t + k + 1/t)\\ &\quad + (C - C_{k+1})\varepsilon(t + k + 1)\end{aligned} \tag{A10}$$

Noting that the second term on the right-hand side contains only the prediction of $n(t+j/t)$ for $j=1,\ldots,k$ and using (A1), it is possible to rewrite this term as

$$\begin{aligned}(D_{k+1}-I)n(t+k+1/t) &= \sum_{j=1}^{k} d_j \left(\frac{C-DE_{k+1-j}}{D}\right)\varepsilon(t+k+1-j)\\ &= \left((D_{k+1}-I)\frac{C}{D}-\sum_{j=1}^{k}(D_{j+1}-D_j)E_{k+1-j}\right)\varepsilon(t+k+1)\\ &= \left((D_{k+1}-I)\frac{C}{D}+E_k\right.\\ &\quad \left.+\sum_{j=0}^{k-1} D_{k+1-j}(E_j-E_{j+1})\right)\varepsilon(t+k+1) \qquad \text{(A11)}\end{aligned}$$

Inserting (A11) into (A10) results in

$$n(t+k+1/t)=\left(\frac{C-DE_{k+1}}{D}\right)\varepsilon(t+k+1) \qquad \text{(A12)}$$

with

$$E_{k+1}=E_k+C_{k+1}+\sum_{j=0}^{k-1} D_{k+1-j}(E_{j+1}-E_{j+2}) \qquad \text{(A13)}$$

which proves that the lemma is equivalent to (A1) by using an iterative formula of eqn. (A2) to calculate E_k.

To complete the proof of the lemma, it must be shown that $C-DE_k$ can be written as

$$C-DE_k=q^{-k}F_k \qquad \text{(A14)}$$

F_k being a polynomial in q^{-1}. This means that the first k coefficients of $C-DE_k$ have to be equal to zero. To make this clear, eqn. (A2) and the condition imposed on eqn. (A14) will be written in matrix form. $\bar{E}_k$ is defined as a column vector containing the coefficients of the polynomial E_k; $\bar{C}_k$ is defined similarly; T_{D_k} is the Toeplitz matrix containing the coefficients of the polynomial D_k:

$$\bar{E}_k=\begin{bmatrix} I \\ e_1 \\ \vdots \\ e_{k-1}\end{bmatrix}, \qquad T_{D_k}=\begin{bmatrix} I & 0 & \ldots & 0 \\ d_1 & I & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ d_{k-1} & d_{k-2} & \ldots & I\end{bmatrix} \qquad \text{(A15)}$$

This means that the statement 'the first k coefficients of $C - DE_k$ are equal to zero' is equivalent to '$\bar{C}_k - T_{D_k}\bar{E}_k = 0$.'

This can easily be seen if we write the recursive form of E_{k+1} (eqn. (A13)) in its matrix form:

$$\begin{bmatrix} \bar{E}_k \\ e_k \end{bmatrix} = \begin{bmatrix} \bar{E}_k \\ 0 \end{bmatrix} + \begin{bmatrix} \bar{C}_k \\ c_k \end{bmatrix} - \begin{bmatrix} T_{D_k} & & 0 \\ d_k & \dots & I \end{bmatrix} \begin{bmatrix} \bar{E}_k \\ 0 \end{bmatrix} \tag{A16}$$

It is clear from (A16) that $\bar{C}_k - T_{D_k}\bar{E}_k = 0$ which proves the correctness of (A14). Combining (A1) with (A14) shows that $n(t + k/t)$ calculated with the lemma is equivalent to using diophantine equations (3). This finishes the proof.

References

Camacho E.F. and Bordons C. (1995): *Model Predictive Control in the Process Industry.* — London: Springer.

Camacho E.F. and Berenguel M. (1994): *Application of generalized predictive control to a solar power plant.* — Proc. 3rd IEEE Conf. *Control Applications*, Glasgow, England, Vol.3, pp.1657–1662.

Chen H., Kremling A. and Allgöwer F. (1995): *Nonlinear predictive control of a benchmark CSTR.* — Proc. 3rd *European Control Conference*, Rome, Italy, Vol.3, pp.3247–3252.

Chen H. and Allgöwer F. (1997): *A quasi-infinite horizon nonlinear predictive control scheme for stable systems: Application to a CSTR.* — IFAC, Int. Symp. *Advanced Control of Chemical Processes*, ADCHEM, Banff, Canada, pp.471–476.

Clarke D.W., Mohtadi C. and Tuffs P.S. (1987): *Generalized predictive control.* — Automatica, Vol.23, No.2, pp.137–161.

Cutler C.R. and Ramaker B.L. (1980): *Dynamic matrix control—A computer control algorithm.* — Proc. Joint American Contr. Conf., San Francisco, pp.WP5–B.

Declercq F. and De Keyser R. (1995): *Using Levenberg-Marquardt minimization in neural model based predictive control.* — Proc. IFAC/IMACS, Int. Workshop *Artificial Intelligence in Real-Time Control*, Bled, Slovenia, pp.275–279.

Declercq F. and De Keyser R. (1997): *A neural model based predictive control toolbox for matlab.* — Proc. 7th IFAC Symp. *Computer Aided Control Systems Design*, BIRA, Gent., pp.305–310.

Demuth H. and Beale M. (1994): *Neural Network Toolbox.* — Natick, Mass: The Mathworks Inc.

Engell S. and Klatt K.-U. (1993): *Nonlinear control of a non-minimum-phase CSTR.* — Proc. *American Control Conference*, San Francisco, Vol.7. pp.2941–2945.

Garcia C.E., Prett D.M. and Morari M. (1989): *Model predictive control: Theory and practice—A survey.* — Automatica, Vol.25, No.3, pp.335–348.

Grace A. (1995): *Optimization Toolbox: For Use with Matlab.* — Natick, Mass: The Math Works, Inc.

Keeler J., Martin G., Boe G., Piche S., Mathur U. and Johnson D. (1996): *The process perfecter: The next step in multivariable control and optimization.* — Technical Report, Pavilion Technologies, Inc.

Lewis F.L. (1986): *Optimal Estimation with an Introduction to Stochastic Control Theory.* — New York: A Wiley-Interscience.

Ljung L. (1987): *System Identification. Theory for the User.* — Londyn: Prentice-Hall.

De Keyser R. (1991): *Basic principles of model based predictive control.* — Proc. 1st *European Control Conference*, Grenoble, France, Vol.3, pp.1753–1758.

De Keyser R. (1998): *A gentle formulation of multivariable model based predictive control.* — Proc. EC-ALFA-PADI2, Int. Conf. *Automatic Control*, Piura, Peru, pp.313–326.

Narendra K.S. and Parthasarathy P. (1990): *Identification and control of dynamic systems using neural networks.* — IEEE Trans. Neural Networks, Vol.1, No.1, pp.4–26.

Norgaard M. (1995): *Neural network based system identification toolbox.* — M.Sc. Thesis, Tech. Rep., No.95-E773, Institute of Automation, Technical University of Denmark, Lyngby.

Ohtsuka T. and Fujii H.A. (1997): *Real-time optimization algorithm for nonlinear receding-horizon control.* — Automatica, Vol.33, No.6, pp.1147–1154.

Richalet J., Rault A., Testud J.L. and Papon J. (1978): *Algorithmic control of industrial processes*, In: IFAC, Identification and System Parameter Estimation (Rajbman, Ed.). — Amsterdam: North-Holland, pp.1119–1167.

Semino D., Di Marco R. and Brambilla A. (1997): *Nonlinear model predictive control with state and parameter estimation.* — Proc. IFAC, Int. Symp. *Advanced Control of Chemical Processes, ADCHEM*, Banff, Canada, pp.505–510.

Sjöberg J., Zhang Q., Ljung L., Benveniste A., Delyon B., Glorennece P.-Y., Hjalmarsson H. and Juditsky A. (1995): *Nonlinear black-box modeling in system identification: A unified overview.* — Automatica, Vol.31, No.12, pp.1691–1724.

Wismer D.A. and Chattergy R. (1978): *Introduction to Non-Linear Optimization. A Problem Solving Approach.* — Amsterdam: North-Holland.