Int. J. Appl. Math. Comput. Sci., 2007, Vol. 17, No. 2, 217–232
DOI: 10.2478/v10006-007-0020-5

amcs

A FAMILY OF MODEL PREDICTIVE CONTROL ALGORITHMS WITH ARTIFICIAL NEURAL NETWORKS

MACIEJ ŁAWRYŃCZUK

Institute of Control and Computation Engineering
Faculty of Electronics and Information Technology
Warsaw University of Technology
ul. Nowowiejska 15/19, 00–665 Warsaw, Poland
e-mail: M.Lawrynczuk@ia.pw.edu.pl

This paper details nonlinear Model-based Predictive Control (MPC) algorithms for MIMO processes modelled by means of neural networks of a feedforward structure. Two general MPC techniques are considered: the one with Nonlinear Optimisation (MPC-NO) and the one with Nonlinear Prediction and Linearisation (MPC-NPL). In the first case a nonlinear optimisation problem is solved in real time on-line. In order to reduce the computational burden, in the second case a neural model of the process is used on-line to determine local linearisation and a nonlinear free trajectory. Single-point and multi-point linearisation methods are discussed. The MPC-NPL structure is far more reliable and less computationally demanding in comparison with the MPC-NO one because it solves a quadratic programming problem, which can be done efficiently within a foreseeable time frame. At the same time, closed-loop performance of both algorithm classes is similar. Finally, a hybrid MPC algorithm with Nonlinear Prediction, Linearisation and Nonlinear optimisation (MPC-NPL-NO) is discussed.

Keywords: predictive control, neural networks, optimisation, linearisation, quadratic programming

1. Introduction

Model predictive control is the only advanced control technique (i.e., more advanced than the well known PID approach) which has been very successful in practical applications. MPC has influenced not only the directions of development of industrial control systems but also research in this field (Brdyś and Tatjewski, 2005; Henson, 1998; Maciejowski, 2002; Morari and Lee, 1999; Qin and Badgwell, 2003; Rossiter, 2003; Tatjewski 2007).The most important advantage of MPC algorithms is the fact that they have the unique ability to take into account constraints imposed on process inputs (manipulated variables) and outputs (controlled variables) or state variables, which usually determine the quality, economic efficiency and safety of production. Furthermore, the MPC technique is very efficient in multivariable process control.

For prediction purposes a dynamic model of the process is used. The choice of the model (a linear model or a nonlinear model; if a nonlinear model—a fundamental model or a black-box model, if a black-box model—its structure) is crucial. This decision affects not only the possible control accuracy but also the computational load and reliability of the whole control policy. When possible, MPC algorithms based on linear models have been applied in practice. In such cases the resulting optimisation problem is a quadratic programming one (Maciejowski, 2002; Morari and Lee 1999; Qin and Badgwell, 2003; Rossiter, 2003; Tatjewski, 2007). Unfortunately, when the process exhibits severe nonlinearity, such an approach is likely to result in poor closed-loop control performance, and even instability. In general, a nonlinear model used for prediction purposes leads to a non-quadratic, non-convex and even multi-modal optimisation problem. For such problems there are no sufficiently fast and reliable optimisation algorithms, i.e., those which would be able to determine the global optimal solution at each sampling instant and within a predefined time limit as required in on-line control. Gradient-based optimisation techniques may terminate in local minima while global ones substantially increase the computational burden, yet they still give no guarantee that the global solution is found (Mahfouf and Linkens, 1998).

In order to overcome the problems inevitable in MPC with nonlinear optimisation, a few alternatives have been

suggested. For example, affine nonlinear models of a neural structure result in a quadratic programming problem (Liu *et al.*, 1998). The computational burden can be significantly reduced when only the first control move is optimised, the remaining ones being obtained using linear MPC (Zheng, 1997). Yet another option is to use a combination of a neural steady-state model and a simplified nonlinear second order quadratic dynamic model (Piche *et al.*, 2000). Although the resulting optimisation task is not convex, the model is relatively simple and the approach is reported to be successful in many industrial applications. For some models, an appropriate structure exploitation (Bloemen *et al.*, 2001) or a change of coordinates (Sriniwas and Arkun, 1997) leads to convexity. An interesting idea is to approximate the nonlinear constrained MPC algorithm by means of a neural network which is trained off-line. During on-line control the manipulated variables are calculated without any optimisation and the neural network replaces the whole MPC algorithm (Åkesson and ToivonenS, 2006; Cavagnari *et al.*, 1999; Parisini *et al.*, 1998). Unfortunately, neural network training is difficult, and hence the approximate MPC approach has limited applicability. A neural network of a specialised structure can also be used, the purpose of which is to solve on-line the MPC optimisation problem (Wang and Wan, 2001). Feedback linearisation is another effective approach to nonlinear MPC (Bacic *et al.*, 2002). It is also possible to linearise a model of the process around a trajectory (Kouvaritakis *et al.*, 1999; Grimble and Ordys, 2001).

Bearing in mind all the aforementioned computational difficulties typical of nonlinear MPC, a straightforward idea is to use linearisation-based MPC techniques, in which only a quadratic programming problem is solved on-line. When compared with MPC algorithms with full nonlinear optimisation, they are suboptimal, but in most practical applications the accuracy is sufficient (Babuška *et al.*, 1999; Henson, 1998; Kavsek *et al.*, 1997; Ławryńczuk, 2003, Ławryńczuk and Tatjewski, 2006; 2003; 2002; Morari and Lee, 1999; Tatjewski and Ławryńczuk, 2006; Tatjewski, 2007). Moreover, one can imagine a combination of the MPC algorithm with linearisation, which determines the initial solution, and the MPC with nonlinear optimisation, which refines the solution. Such an approach has the advantages of both structures, i.e., computational efficiency and accuracy.

In light of practical implementation, the main issue to address is the choice of the process model structure, since it affects the performance and accuracy of the control algorithm. Fundamental (first-principle) models, although potentially very precise, are usually not suitable for on-line control since they are very complicated and may lead to numerical problems, e.g., ill-conditioning. As far as empirical models are concerned, feedforward neural network models deserve attention because they have the following advantages:

(a) they constitute universal approximators (Hornik *et al.*, 1989), and hence are able to approximate precisely nonlinear behaviours of technological dynamic processes (Hussain, 1999; Nørgaard *et al.*, 2000; Piche *et al.*, 2000),

(b) efficient identification (training) algorithms and structure optimisation techniques have been developed (Haykin, 1999; Osowski, 1996),

(c) they have a relatively small number of parameters (unlike fuzzy models, they do not suffer from the "curse of dimensionality") and simple structures,

(d) they can be easily incorporated into MPC algorithms and efficiently used on-line (Hussain, 1999; Ławryńczuk, 2003; Ławryńczuk and Tatjewski, 2006; 2003; 2002; 2001; Nørgaard *et al.*, 2000; Tatjewski, 2007; Tatjewski and Ławryńczuk, 2006; Trajanoski and Wach, 1998; Yu and Gomm, 2003).

The outline of the paper is as follows: First, Section 2 states the MPC optimisation problem. In Section 3 the structure of the neural model is defined. Section 4 details the MPC algorithm with Nonlinear Optimisation (MPC-NO), while Section 5 describes suboptimal MPC techniques with Nonlinear Prediction and Linearisation (MPC-NPL). Two linearisation methods are then considered. Section 6 deals with a hybrid MPC algorithm with Nonlinear Prediction, Linearisation and Nonlinear optimisation (MPC-NPL-NO). Simulation results of these algorithms applied to two nonlinear processes (an SISO polymerisation reactor and an MIMO distillation column) are presented in Section 7. The paper is summarised in Section 8.

2. Model Predictive Control

Although a number of different MPC algorithms have been developed over the years, the main idea (i.e., the explicit application of a process model, the receding horizon and optimisation of a cost function) is always the same (Brdyś and Tatjewski, 2005; Maciejowski, 2002; Rossiter, 2003; Tatjewski 2007). At each consecutive sampling instant k a set of future controls or corresponding increments

$$\mathbf{u}(k) = \begin{bmatrix} u(k|k) \\ \vdots \\ u(k+N_u-1|k) \end{bmatrix},$$

$$\Delta\mathbf{u}(k) = \begin{bmatrix} \Delta u(k|k) \\ \vdots \\ \Delta u(k+N_u-1|k) \end{bmatrix} \tag{1}$$

is determined, where N_u is the control horizon. It is assumed that $u(k+p|k) = u(k+N_u-1|k)$ for $p \geq N_u$. The

decision variables of the MPC algorithm, $\mathbf{u}(k)$ or $\Delta\mathbf{u}(k)$, are calculated so as to minimise the differences between the predicted values of the outputs (or states) and the reference trajectory over the prediction horizon. Only the first element of the determined sequence is applied to the process, so that the control law is

$$u(k) = u(k|k) \quad \text{or} \quad u(k) = \Delta u(k|k) + u(k-1). \tag{2}$$

At the next sampling instant, $k+1$, the measurement of the process output (or state) variables is updated, the prediction is shifted one step forward and the whole procedure is repeated.

2.1. Model Predictive Control Optimisation Problem. In the MPC dynamic optimisation problem, the predicted values of the control errors over the prediction horizon N and future control moves over the control horizon, N_u, are minimised,

$$\min_{\mathbf{u}(k)}\Big\{J(k) = \sum_{p=1}^{N} \big\|y^{\mathrm{ref}}(k+p|k) - \hat{y}(k+p|k)\big\|^2_{\mathbf{M}_p} + \sum_{p=0}^{N_u-1} \|\Delta\mathbf{u}(k+p|k)\|^2_{\mathbf{\Lambda}_p}\Big\},$$

subject to

$$\begin{aligned} u_{\min} \le\ & u(k+p|k) \le\ u_{\max},\ p=0,\ldots,N_u-1,\\ -\Delta u_{\max} \le\ & \Delta u(k+p|k) \le \Delta u_{\max},\ p=0,\ldots,N_u-1,\\ y_{\min} \le\ & \hat{y}(k+p|k) \le\ y_{\max},\ p=1,\ldots,N, \end{aligned} \tag{3}$$

where $\mathbf{M_p} \ge \mathbf{0}$ and $\mathbf{\Lambda}_p > 0$ are diagonal weighting matrices with dimensions $n_y \times n_y$ and $n_u \times n_u$, respectively, $\hat{y}(k+p|k)$ denotes the prediction of the outputs for the future sampling instant $k+p$ calculated at the current sampling instant k using a dynamic model of the process. The reference trajectory, $y^{\mathrm{ref}}(k+p|k)$, is typically assumed to be constant over the prediction horizon and equal to the desired set-point, i.e.,

$$y^{\mathrm{ref}}(k+p|k) = y^{\mathrm{sp}}(k), \qquad p=1,\ldots,N. \tag{4}$$

If the process exhibits a significant time-delay, it is reasonable to summarise the predicted control errors in the first part of the cost-function $J(k)$ commencing with $p = N_1 > 1$.

If the output constraints have to be taken into account, the controller may be affected by the infeasibility problem. In order to cope with such a situation, the well-known approach is to soften the output constraints by using slack variables (Maciejowski, 2002). Using a quadratic penalty for constraint violations, the MPC optimisation problem (3) becomes

$$\min_{\mathbf{u}(k),\,\varepsilon_{\min},\,\varepsilon_{\max}} \Big\{J(k) = \big\|\mathbf{y}^{\mathrm{ref}}(k) - \hat{\mathbf{y}}(k)\big\|^2_{\mathbf{M}} + \big\|\Delta\mathbf{u}(k)\big\|^2_{\mathbf{\Lambda}} + \rho_{\min}\big\|\varepsilon_{\min}\big\|^2 + \rho_{\max}\big\|\varepsilon_{\max}\big\|^2\Big\}$$

subject to

$$\begin{aligned} \boldsymbol{u}_{\min} \le \boldsymbol{u}(k) \le \boldsymbol{u}_{\max},\\ -\Delta\boldsymbol{u}_{\max} \le \Delta\boldsymbol{u}(k) \le \Delta\boldsymbol{u}_{\max},\\ \boldsymbol{y}_{\min} - \boldsymbol{\varepsilon}_{\min} \le \hat{y}(k) \le \boldsymbol{y}_{\max} + \boldsymbol{\varepsilon}_{\max},\\ \boldsymbol{\varepsilon}_{\min} \ge 0, \quad \boldsymbol{\varepsilon}_{\max} \ge 0, \end{aligned} \tag{5}$$

where

$$\mathbf{u}_{\min} = \begin{bmatrix} u_{\min} \\ \vdots \\ u_{\min} \end{bmatrix}, \quad \mathbf{u}_{\max} = \begin{bmatrix} u_{\max} \\ \vdots \\ u_{\max} \end{bmatrix}, \quad \Delta\mathbf{u}_{\max} = \begin{bmatrix} \Delta u_{\max} \\ \vdots \\ \Delta u_{\max} \end{bmatrix} \tag{6}$$

are vectors of length $n_u N_u$, and

$$\mathbf{y}^{\mathrm{ref}}(k) = \begin{bmatrix} y^{\mathrm{ref}}(k+1|k) \\ \vdots \\ y^{\mathrm{ref}}(k+N|k) \end{bmatrix}, \quad \hat{\mathbf{y}}(k) = \begin{bmatrix} \hat{y}(k+1|k) \\ \vdots \\ \hat{y}(k+N|k) \end{bmatrix}, \quad \mathbf{y}_{\min} = \begin{bmatrix} y_{\min} \\ \vdots \\ y_{\min} \end{bmatrix}, \quad \mathbf{y}_{\max} = \begin{bmatrix} y_{\max} \\ \vdots \\ y_{\max} \end{bmatrix} \tag{7}$$

are vectors of length $n_y N$. Diagonal weighting matrices $\mathbf{M}$ and $\mathbf{\Lambda}$, of dimensionality $n_y N \times n_y N$ and $n_u N_u \times n_u N_u$, respectively, are

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_p & & \\ & \ddots & \\ & & \mathbf{M}_p \end{bmatrix}, \quad \mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}_p & & \\ & \ddots & \\ & & \mathbf{\Lambda}_p \end{bmatrix}. \tag{8}$$

In the MPC optimisation problem (5), $\varepsilon_{\min}$ and $\varepsilon_{\max}$ are vectors of length $n_y N$ containing slack variables, and $\rho_{\min}$, $\rho_{\max}$ are positive weights.

3. Structure of the Neural Model

The model of the Multi-Input Multi-Output (MIMO) process under consideration is comprised of n_y Multi-Input Single-Output (MISO) models, where n_y is the number of outputs. Let each consecutive MISO model be described by the following nonlinear discrete-time equation:

$$\begin{aligned} y_m(k) = g_m(&u_1(k-\tau^{m,1}),\ldots,u_1(k-n_b^{m,1}),\ldots,\\ &u_{n_u}(k-\tau^{m,n_u}),\ldots u_{n_u}(k-n_b^{m,n_u}),\\ &y_m(k-1),\ldots,y_m(k-n_a^m)), \end{aligned} \tag{9}$$

where

$$g_m : \mathbb{R}^{n_a^m+\sum_{n=1}^{n_u}(n_b^{m,n}-\tau^{m,n}+1)} \longrightarrow \mathbb{R} \in C^1,$$

$m=1,\ldots,n_y$, $\tau^{m,n} \le n_{n_b}^{m,n}$, n_u is the number of inputs. As the model of the process, n_y feedforward neural networks with one nonlinear hidden layer and linear output (Haykin, 1999; Osowski, 1996) are used. The structure of the neural model of the m-th output is depicted in Fig. 1.

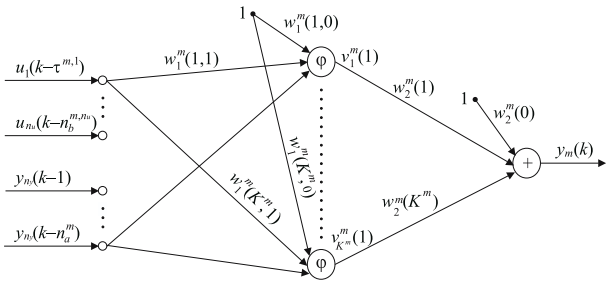

Fig. 1. Structure of the neural model of the m-th output.

The output of the model can be expressed as

$$\begin{aligned} y_m(k) &= w_2^m(0)+\sum_{i=1}^{K^m} w_2^m(i)v_i^m(k)\\ &= w_2(0)+\sum_{i=1}^{K^m} w_2^m(i)\varphi\Big(z_i^m(k)\Big), \end{aligned} \tag{10}$$

where $z_i^m(k)$ and $v_i^m(k)$ are the sum of inputs and the output of the i-th hidden node, respectively, $\varphi : \mathbb{R} \longrightarrow \mathbb{R}$ is the nonlinear transfer function (e.g., hyperbolic tangent), K^m is the number of nonlinear hidden nodes. Recalling the input arguments of the general nonlinear model (9), we have

$$\begin{aligned} z_i^m(k) &= w_1^m(i,0)\\ &\quad+\sum_{n=1}^{n_u}\sum_{j=1}^{I_u^{m,n}} w_1^m(i,R^{m,n}+j)u_n(k-\tau^{m,n}+1-j)\\ &\quad+\sum_{j=1}^{n_a^m} w_1^m(i,S^m+j)y_m(k-j). \end{aligned} \tag{11}$$

The weights of the m-th network are denoted by $w_1^m(i,j)$, $i=1,\ldots,K^m$, $j=0,\ldots,n_a^m+\sum_{n=1}^{n_u}(n_b^{m,n}-\tau^{m,n}+1)$, and $w_2^m(i)$, $i=0,\ldots,K^m$, for the first and second layers, respectively. The number of the network's input nodes depending on input signals u_n, $n=1,\ldots,n_u$ is $I_u^{m,n}=n_b^{m,n}-\tau^{m,n}+1$. The total number of weights is $(n_a^m+\sum_{n=1}^{n_u}(n_b^{m,n}-\tau^{m,n}+1)+1)K^m+K^m+1$. The auxiliary coefficients are

$$R^{m,n} = \begin{cases} 0 & \text{if } n=1,\\ \sum_{i=1}^{n-1} I_u^{m,i} & \text{if } n=2,\ldots,n_u, \end{cases} \tag{12a}$$

$$S^m = \sum_{i=1}^{n_u} I_u^{m,i}. \tag{12b}$$

The control algorithms described in this paper use input-output neural models of processes although a state-space representation may be necessary in some cases (Dutka and Ordys, 2004; Grimble and Ordys, 2001). It is assumed that sufficiently large data sets can be collected which are next used in the off-line training of the neural model. When necessary, on-line model adaptation should be used.

4. MPC Algorithm with Nonlinear Optimisation (MPC–NO)

4.1. MPC-NO Optimisation Problem. In the MPC-NO algorithm, the nonlinear neural model is used for prediction purposes. At each sampling instant, future values of control signals, $\mathbf{u}(k)$, are determined as the solution to a nonlinear optimisation problem. The structure of the MPC-NO algorithm is depicted in Fig. 2. From (5), the MPC-NO optimisation problem is

$$\begin{aligned} \min_{\mathbf{u}(k),\,\varepsilon_{\min},\varepsilon_{\max}} \Big\{ J(k) = &\|\mathbf{y}^{\mathrm{ref}}(k)-\hat{\mathbf{y}}(k)\|_{\mathbf{M}}^2\\ &+\|\mathbf{J}^{\mathrm{NO}}\mathbf{u}(k)+\mathbf{u}^{\mathrm{NO}}(k)\|_{\mathbf{\Lambda}}^2\\ &+\rho_{\min}\|\varepsilon_{\min}\|^2+\rho_{\max}\|\varepsilon_{\max}\|^2\Big\} \end{aligned}$$

subject to

$$\begin{aligned} \mathbf{u}_{\min} \le \mathbf{u}(k) \le \mathbf{u}_{\max}\\ -\Delta\mathbf{u}_{\max} \le \mathbf{J}^{\mathrm{NO}}\mathbf{u}(k)+\mathbf{u}^{\mathrm{NO}}(k) \le \Delta\mathbf{u}_{\max}\\ \mathbf{y}_{\min}-\varepsilon_{\min} \le \hat{\mathbf{y}}(k) \le \mathbf{y}_{\max}+\varepsilon_{\max}\\ \varepsilon_{\min} \ge 0, \qquad \varepsilon_{\max} \ge 0 \end{aligned} \tag{13}$$

where $\mathbf{J}^{\mathrm{NO}}$ is an $n_uN_u \times n_uN_u$ matrix, $\mathbf{u}^{NO}(k)$ is

an $n_u N_u$-dimensional vector,

$$\mathbf{J}^{\mathrm{NO}} = \begin{bmatrix} \mathbf{I}_{n_u \times n_u} & \mathbf{0}_{n_u \times n_u} & \mathbf{0}_{n_u \times n_u} & \dots & \mathbf{0}_{n_u \times n_u} \\ -\mathbf{I}_{n_u \times n_u} & \mathbf{I}_{n_u \times n_u} & \mathbf{0}_{n_u \times n_u} & \dots & \mathbf{0}_{n_u \times n_u} \\ \mathbf{0}_{n_u \times n_u} & -\mathbf{I}_{n_u \times n_u} & \mathbf{I}_{n_u \times n_u} & \dots & \mathbf{0}_{n_u \times n_u} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{n_u \times n_u} & \mathbf{0}_{n_u \times n_u} & \mathbf{0}_{n_u \times n_u} & \dots & \mathbf{I}_{n_u \times n_u} \end{bmatrix},$$

$$\mathbf{u}^{\mathrm{NO}}(k) = \begin{bmatrix} -u(k-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{14}$$

Here $\mathbf{I}_{n_u \times n_u}$ and $\mathbf{0}_{n_u \times n_u}$ are $n_u \times n_u$ identity and zero matrices, respectively.

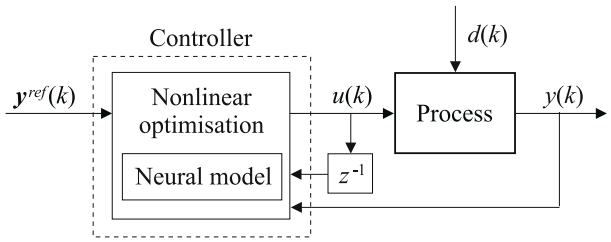

Fig. 2. Structure of the MPC algorithm with Nonlinear Optimisation (MPC-NO). Here $d(k)$ stands for the unmeasured disturbance.

As regards the computational burden of the MPC-NO optimisation problem (13), a fundamental issue is a proper choice of the initial point, $\mathbf{u}^0(k)$, which would result in fast convergence of the nonlinear optimisation subroutine. To use a constant vector is not an effective approach, since it is independent of the current operating point. It is much better to use the values of the manipulated variables calculated and applied to the plant at the previous sampling instant, i.e., $\mathbf{u}^0(k) = [u(k-1) \dots u(k-1)]^T$. Alternatively, one may use the last $n_u(N_u - 1)$ control values calculated at the previous sampling instant and not applied to the process, i.e.,

$$\mathbf{u}^0(k) = \begin{bmatrix} u^0(k|k) \\ \vdots \\ u^0(k+N_u-3|k) \\ u^0(k+N_u-2|k) \\ u^0(k+N_u-1|k) \end{bmatrix} = \begin{bmatrix} u(k|k-1) \\ \vdots \\ u(k+N_u-3|k-1) \\ u(k+N_u-2|k-1) \\ u(k+N_u-2|k-1) \end{bmatrix}. \tag{15}$$

The initial values of the slack variables $\varepsilon_{\min}$, $\varepsilon_{\max}$ are set to zero.

4.2. Calculation of Gradients. In general, one can imagine two methods of using the neural model given by (10) and (11) in the MPC-NO scheme with nonlinear optimisation. In the first approach, the gradients of the cost function $J(k)$ are approximated numerically and the nonlinear optimisation problem (13) is solved on-line (Hussain, 1999; Trajanoski and Wach, 1998; Yu and Gomm, 2003). In the second approach, the structure of the neural model is exploited (Ławryńczuk, 2003; Ławryńczuk and Tatjewski, 2001; Nørgaard *et al.*, 2000; Tatjewski, 2007; Tatjewski and Ławryńczuk, 2006). The latter approach is recommended in this paper.

Differentiating the cost function $J(k)$ with respect to the future control sequence, $\mathbf{u}(k)$, results in

$$\frac{\mathrm{d}J(k)}{\mathrm{d}\mathbf{u}(k)} = 2\left(\frac{\mathrm{d}\hat{\mathbf{y}}(k)}{\mathrm{d}\mathbf{u}(k)}\right)^T \mathbf{M}(\hat{\mathbf{y}}(k) - \mathbf{y}^{\mathrm{ref}}(k)) + 2(\mathbf{J}^{\mathrm{NO}})^T \mathbf{\Lambda}(\mathbf{J}^{\mathrm{NO}}\mathbf{u}(k) + \mathbf{u}^{NO}(k)). \tag{16}$$

The $n_y N \times n_u N_u$ matrix of the partial derivatives of the predicted outputs with respect to future controls is

$$\frac{\mathrm{d}\hat{\mathbf{y}}(k)}{\mathrm{d}\mathbf{u}(k)} = \begin{bmatrix} \frac{\mathrm{d}\hat{y}(k+1|k)}{\mathrm{d}u(k|k)} & \dots & \frac{\mathrm{d}\hat{y}(k+1|k)}{\mathrm{d}u(k+N_u-1|k)} \\ \vdots & \ddots & \vdots \\ \frac{\mathrm{d}\hat{y}(k+N|k)}{\mathrm{d}u(k|k)} & \dots & \frac{\mathrm{d}\hat{y}(k+N|k)}{\mathrm{d}u(k+N_u-1|k)} \end{bmatrix}, \tag{17}$$

where

$$\frac{\mathrm{d}\hat{y}(k+p|k)}{\mathrm{d}u(k+r|k)} = \begin{bmatrix} \frac{\mathrm{d}\hat{y}_1(k+p|k)}{\mathrm{d}u_1(k+r|k)} & \dots & \frac{\mathrm{d}\hat{y}_1(k+p|k)}{\mathrm{d}u_{n_u}(k+r|k)} \\ \vdots & \ddots & \vdots \\ \frac{\mathrm{d}\hat{y}_{n_y}(k+p|k)}{\mathrm{d}u_1(k+r|k)} & \dots & \frac{\mathrm{d}\hat{y}_{n_y}(k+p|k)}{\mathrm{d}u_{n_u}(k+r|k)} \end{bmatrix}, \tag{18}$$

are $n_y \times n_u$ submatrices for all $p = 1, \dots, N$, $r = 0, \dots, N_u - 1$. The predictions $\hat{y}_m(k+p|k)$ for $m = 1, \dots, n_y$, $p = 1, \dots, N$ are calculated from the general prediction equation (Maciejowski, 2002; Tatjewski, 2007),

$$\hat{y}_m(k+p|k) = y_m(k+p|k) + d_m(k), \tag{19}$$

where the quantities $y_m(k+p|k)$ are calculated from the neural model given by (10) and (11) applied to the sampling instant $k+p$ at the current sampling instant k. The above formulation uses the "DMC type" disturbance model, in which the unmeasured disturbance $d_m(k)$ is assumed to be constant over the prediction horizon. Its value

is estimated from the equation

$$d_m(k) = y_m(k) - y_m(k|k-1) = y_m(k) - \left(w_2^m(0) + \sum_{i=1}^{K^m} w_2^m(i)\varphi(z_i^m(k)) \right), \quad (20)$$

where

$$y_m(k+p|k) = w_2^m(0) + \sum_{i=1}^{K^m} w_2^m(i)\varphi(z_i^m(k+p|k)). \quad (21)$$

As regards the prediction of the m-th output over the horizon N for the sampling instant $k+p$ computed at the current sampling instant k, the quantities $z_i^m(k+p|k)$ and, consequently, $y_m(k+p|k)$ depend on some control signal values applied to the plant at previous sampling instants, future control signals (i.e., decision variables of the control algorithm), measured values of the plant output signal and future output predictions. From (11) one has

$$\begin{aligned} z_i^m(k+p|k) &= w_1^m(i,0) + \sum_{n=1}^{n_u} \sum_{j=1}^{I_{uf}^{m,n}(p)} w_1^m(i, R^{m,n}+j) \\ &\quad \times u_n(k-\tau^{m,n}+1-j+p|k) \\ &\quad + \sum_{n=1}^{n_u} \sum_{j=I_{uf}^{m,n}(p)+1}^{I_u^{m,n}} w_1^m(i, R^{m,n}+j) \\ &\quad \times u_n(k-\tau^{m,n}+1-j+p) \\ &\quad + \sum_{j=1}^{I_{yp}^m(p)} w_1^m(i, S^m+j)\hat{y}_m(k-j+p|k) \\ &\quad + \sum_{j=I_{yp}^m(p)+1}^{n_a^m} w_1^m(i, S^m+j) y_m(k-j+p), \quad (22) \end{aligned}$$

where $I_{uf}^{m,n}(p) = \max\{\min\{p-\tau^{m,n}+1, I_u^{m,n}\}, 0\}$ is the number of the m-th network's input nodes depending on future control signals of the n-th input and $I_{yp}^m(p) = \min\{p-1, n_a^m\}$ is the number of the m-th network's input nodes depending on output predictions. Because typically $N_u < N$ (hence $u_n(k+p|k) = u_n(k+N_u-1|k)$ for $p \geq N_u$), it can be noticed that

$$\begin{aligned} z_i^m(k+p|k) &= w_1^m(i,0) + \sum_{n=1}^{n_u} \sum_{j=1}^{I_{N_u}^{m,n}(p)} w_1^m(i, R^{m,n}+j) \\ &\quad \times u_n(k+N_u-1|k) \\ &\quad + \sum_{n=1}^{n_u} \sum_{j=I_{N_u}^{m,n}(p)+1}^{I_{uf}^{m,n}(p)} w_1^m(i, R^{m,n}+j) \\ &\quad \times u_n(k-\tau^{m,n}+1-j+p|k) \\ &\quad + \sum_{n=1}^{n_u} \sum_{j=I_{uf}^{m,n}(p)+1}^{I_u^{m,n}} w_1^m(i, R^{m,n}+j) \\ &\quad \times u_n(k-\tau^{m,n}+1-j+p) \\ &\quad + \sum_{j=1}^{I_{yp}^m(p)} w_1^m(i, S^m+j)\hat{y}_m(k-j+p|k) \\ &\quad + \sum_{j=I_{yp}^m(p)+1}^{n_a^m} w_1^m(i, S^m+j) y_m(k-j+p), \quad (23) \end{aligned}$$

where $I_{N_u}^{m,n}(p) = \min\{\max\{p-\tau^{m,n}-N_u+1, 0\}, I_u^{m,n}\}$ is the number of the m-th network's input nodes depending on the quantity $u_n(k+N_u-1|k)$.

Taking into account (19) and (21), the entries of the matrix $\mathrm{d}\hat{\mathbf{y}}(k)/\mathrm{d}\mathbf{u}(k)$, i.e., the partial derivatives of the predicted output signal with respect to future controls are determined from

$$\frac{\mathrm{d}\hat{y}_m(k+p|k)}{\mathrm{d}u_n(k+r|k)} = \sum_{i=1}^{K^m} w_2^m(i) \frac{\mathrm{d}\varphi(z_i^m(k+p|k))}{\mathrm{d}z_i^m(k+p|k)} \frac{\mathrm{d}z_i^m(k+p|k)}{\mathrm{d}u_n(k+r|k)}. \quad (24)$$

Obviously,

$$\frac{\mathrm{d}z_i^m(k+p|k)}{\mathrm{d}u_n(k+r|k)} = \frac{\mathrm{d}\hat{y}_m(k+p|k)}{\mathrm{d}u_n(k+r|k)} = 0, \quad r \geq p-\tau^{m,n}+1. \quad (25)$$

If the hyperbolic tangent is used as the nonlinear transfer function φ in the hidden layer of the neural model, one has

$$\frac{\mathrm{d}\varphi(z_i^m(k+p|k))}{\mathrm{d}z_i^m(k+p|k)} = 1 - \tanh^2(z_i^m(k+p|k)).$$

It can be noticed that decision variables of the algorithm affect only the first, second and fourth sums in (23). What is more, only some of the output predictions are influenced by future controls. Hence

$$\begin{aligned} \frac{\mathrm{d}z_i^m(k+p|k)}{\mathrm{d}u_n(k+r|k)} &= \sum_{j=1}^{I_{N_u}^{m,n}(p)} w_1^m(i, R^{m,n}+j) \frac{\mathrm{d}u_n(k+N_u-1|k)}{\mathrm{d}u_n(k+r|k)} \\ &\quad + \sum_{j=I_{N_u}^{m,n}(p)+1}^{I_{uf}^{m,n}(p)} w_1^m(i, R^{m,n}+j) \\ &\quad \times \frac{\mathrm{d}u_n(k-\tau^{m,n}+1-j+p|k)}{\mathrm{d}u_n(k+r|k)} \\ &\quad + \sum_{j=1}^{I_{ypf}^m(p)} w_1^m(i, S^m+j) \frac{\mathrm{d}\hat{y}_m(k-j+p|k)}{\mathrm{d}u_n(k+r|k)}, \quad (26) \end{aligned}$$

where $I_{\mathrm{ypf}}(p) = \max\{\min\{p - \tau^{m,n}, n_a^m\}, 0\}$ is the number of the m-th network's input nodes depending on output predictions which are affected by future controls of the n-th input.

The discussed method of computing the gradients of the predicted output trajectory with respect to the future controls is used not only for obtaining the gradients of the cost function $J(k)$ but also for finding gradients of output constraints. Sequential Quadratic Programming (SQP) (Bazaraa *et al.*, 1993) is used for solving the nonlinear MPC-NO optimisation problem (13). Although an analytical Hessian matrix can be used in the SQP algorithm implementation, it requires much more computational effort than computing the gradients. That is why in the presented solution the optimisation routine is provided with analytical gradients while the Hessian is approximated, as is done in most SQP implementations.

5. MPC Algorithms with Nonlinear Prediction and Linearisation (MPC-NPL)

5.1. MPC-NPL Optimisation Problem. The idea of the MPC-NPL algorithm consists in taking advantage of on-line linearisation and nonlinear free trajectory prediction. More specifically, at each sampling instant k, taking into account the current state of the plant, the model of the process is linearised on-line and a nonlinear free trajectory is determined. Analogously to MPC algorithms with linear models, e.g., DMC (Cutler and Ramaker, 1979) or GPC (Clarke *et al.*, 1987), it is assumed that the output prediction can be expressed as the sum of a forced trajectory, which depends only on the future, i.e., on the input moves $\Delta\mathbf{u}(k)$, and a free trajectory $\mathbf{y}^0(k)$, which depends only on the past. One has

$$\hat{\mathbf{y}}(k) = \mathbf{G}(k)\Delta\mathbf{u}(k) + \mathbf{y}^0(k), \tag{27}$$

where

$$\mathbf{G}(k) = \begin{bmatrix} \mathbf{S}_1(k) & \mathbf{0}_{n_y \times n_u} & \dots & \mathbf{0}_{n_y \times n_u} \\ \mathbf{S}_2(k) & \mathbf{S}_1(k) & \dots & \mathbf{0}_{n_y \times n_u} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_N(k) & \mathbf{S}_{N-1}(k) & \dots & \mathbf{S}_{N-N_u+1}(k) \end{bmatrix} \tag{28}$$

is a dynamic $n_y N \times n_u N_u$ matrix which is composed of step-response coefficients of the linearised model of the process. For the discussed MIMO process having n_u inputs and n_y outputs, the step-response submatrices are

$$\mathbf{S}_j(k) = \begin{bmatrix} s_j^{1,1}(k) & \dots & s_j^{1,n_u}(k) \\ \vdots & \ddots & \vdots \\ s_j^{n_y,1}(k) & \dots & s_j^{n_y,n_u}(k) \end{bmatrix}, \tag{29}$$

and the free trajectory vector is

$$\mathbf{y}^0(k) = \begin{bmatrix} y^0(k+1|k) \\ \vdots \\ y^0(k+N|k) \end{bmatrix}. \tag{30}$$

Of course, the plant is nonlinear and the superposition principle (27) cannot be exactly satisfied at each sampling instant as is the case in linear MPC techniques. In other words, the suboptimal prediction obtained from (27) is different from that determined by means of a nonlinear model, as is done in the MPC-NO algorithm. Nevertheless, taking into account (27), the nonlinear optimisation problem (13) solved in the MPC-NO algorithm becomes the following quadratic programming one:

$$\min_{\Delta\mathbf{u}(k),\, \varepsilon_{\min},\, \varepsilon_{\max}} \left\{ \begin{array}{l} J(k) \\ \quad = \|\mathbf{y}^{\mathrm{ref}}(k) - \mathbf{G}(k)\Delta\mathbf{u}(k) \\ \quad - \mathbf{y}^0(k)\|^2_{\mathbf{M}} + \|\Delta\mathbf{u}(k)\|^2_{\mathbf{\Lambda}} \\ \quad + \rho_{\min}\|\varepsilon_{\min}\|^2 + \rho_{\max}\|\varepsilon_{\max}\|^2 \end{array} \right\}$$

subject to

$$\begin{gathered} \mathbf{u}_{\min} \le \mathbf{J}^{NPL}\Delta\mathbf{u}(k) + \mathbf{u}^{NPL}(k) \le \mathbf{u}_{\max}, \\ -\Delta\mathbf{u}_{\max} \le \Delta\mathbf{u}(k) \le \Delta\mathbf{u}_{\max}, \\ \mathbf{y}_{\min} - \varepsilon_{\min} \le \mathbf{G}(k)\Delta\mathbf{u}(k) + \mathbf{y}^0(k) \le \mathbf{y}_{\max} + \varepsilon_{\max}, \\ \varepsilon_{\min} \ge 0, \qquad \varepsilon_{\max} \ge 0 \end{gathered} \tag{31}$$

where $\mathbf{J}^{\mathrm{NPL}}$ is an $n_u N_u \times n_u N_u$ matrix and $\mathbf{u}^{\mathrm{NPL}}(k)$ is an $n_u N_u$ dimensional vector,

$$\mathbf{J}^{\mathrm{NPL}} = \begin{bmatrix} \mathbf{I}_{n_u \times n_u} & \mathbf{0}_{n_u \times n_u} & \mathbf{0}_{n_u \times n_u} & \dots & \mathbf{0}_{n_u \times n_u} \\ \mathbf{I}_{n_u \times n_u} & \mathbf{I}_{n_u \times n_u} & \mathbf{0}_{n_u \times n_u} & \dots & \mathbf{0}_{n_u \times n_u} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{I}_{n_u \times n_u} & \mathbf{I}_{n_u \times n_u} & \mathbf{I}_{n_u \times n_u} & \dots & \mathbf{I}_{n_u \times n_u} \end{bmatrix},$$

$$\mathbf{u}^{\mathrm{NPL}}(k) = \begin{bmatrix} u(k-1) \\ \vdots \\ u(k-1) \end{bmatrix}. \tag{32}$$

The structure of the MPC-NPL algorithm is depicted in Fig. 3. At each sampling instant k the following steps are repeated:

1. Linearisation of the nonlinear neural model: obtain the dynamic matrix $\mathbf{G}(k)$.
2. Compute the nonlinear free trajectory $\mathbf{y}^0(k)$ using the nonlinear neural model.

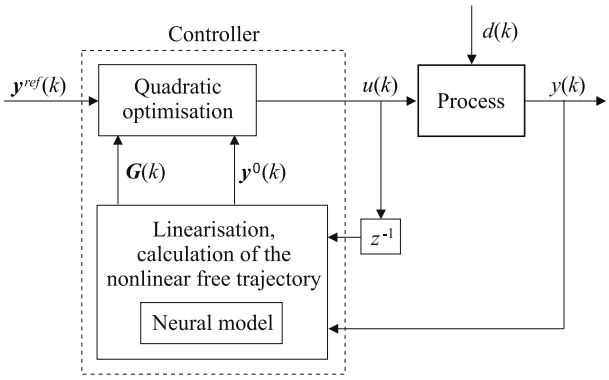

Fig. 3. Structure of the MPC algorithm with Nonlinear Prediction and Linearisation (MPC-NPL). Here $d(k)$ constitutes the unmeasured disturbance.

3. Solve the quadratic programming problem (31) to determine $\Delta\mathbf{u}(k)$.
4. Apply $u(k) = \Delta u(k|k) + u(k-1)$.
5. Set $k := k+1$, go to Step 1.

5.2. On-Line Linearisation of the Neural Model.

5.2.1. Single-Point Linearisation. Defining n_y linearisation points as vectors of length $n_a^m + \sum_{n=1}^{n_u}(n_b^{m,n} - \tau^{m,n}+1)$ composed of past input and output values corresponding to the arguments of the nonlinear model (9) used for the sampling instant $k+1$,

$$\begin{aligned}\overline{x}_m(k) &= \Big[\overline{u}_1(k-\tau^{m,1}+1)\ \ldots\ \overline{u}_1(k-n_b^{m,1}+1)\ \ldots \\ &\quad \overline{u}_{n_u}(k-\tau^{m,n_u}+1)\ \ldots\ \overline{u}_{n_u}(k-n_b^{m,n_u}+1) \\ &\quad \overline{y}_m(k)\ \ldots\ \overline{y}_m(k-n_a^m+1)\Big]^T,\end{aligned} \tag{33}$$

where $m = 1,\ldots,n_y$, and using a Taylor series expansion at these points, the linearised model has the form

$$\begin{aligned}y_m(k) &= g_m\Big(\overline{x}_m(k)\Big) + \sum_{n=1}^{n_u}\sum_{l=1}^{n_b} b_l^{m,n}\Big(\overline{x}_m(k)\Big)(u_n(k-l) \\ &\quad - \overline{u}_n(k-l+1)) \\ &\quad - \sum_{l=1}^{n_a} a_l^m\Big(\overline{x}_m(k)\Big)\Big(y_m(k-l) - \overline{y}_m(k-l+1)\Big),\end{aligned} \tag{34}$$

where

$$\begin{aligned}a_l^m(\overline{x}(k)) &= -\frac{\mathrm{d}g^m(\overline{x}(k))}{\mathrm{d}y_m(k-l)}, \\ b_l^{m,n}(\overline{x}(k)) &= \frac{\mathrm{d}g_m(\overline{x}(k))}{\mathrm{d}u_n(k-l)}.\end{aligned} \tag{35}$$

If $\tau^{m,n} = 1$, then for linearisation purposes one may set $\overline{u}_n(k) = u_n(k-1)$ or $\overline{u}_n(k) = u_n(k|k-1)$.

Taking into account the structure of the neural model and the corresponding equations (10) and (11), the coefficients of the linearised model are calculated from

$$\begin{aligned}&a_l^m(\overline{x}(k)) \\ &= \begin{cases} -\sum_{i=1}^{K^m} w_2^m(i)\frac{\mathrm{d}\varphi(z_i^m(\overline{x}_m(k)))}{\mathrm{d}z_i^m(\overline{x}_m(k))} w_1^m(i, S^m+l) \\ \qquad \text{if } \ l = 1,\ldots,n_a^m, \\ 0 \qquad \text{if } \ l = n_a^m+1,\ldots,n_a \end{cases}\end{aligned} \tag{36}$$

for all $m = 1,\ldots,n_y,\ \ l = 1,\ldots,n_a$, and

$$\begin{aligned}&b_l^{m,n}(\overline{x}(k)) \\ &= \begin{cases} 0 \qquad \text{if } \ l = 1,\ldots,\tau^{m,n}-1, \\ \sum_{i=1}^{K} w_2^m(i)\frac{\mathrm{d}\varphi(z_i^m(\overline{x}_m(k)))}{\mathrm{d}z_i^m(\overline{x}_m(k))} \\ \times w_1^m(i, R^{m,n}+l-\tau^{m,n}+1) \\ \qquad \text{if } \ l = \tau^{m,n},\ldots,n_b^{m,n}, \\ 0 \qquad \text{if } \ l = n_b^{m,n}+1,\ldots,n_b \end{cases}\end{aligned} \tag{37}$$

for all $m = 1,\ldots,n_y$, $n = 1,\ldots,n_u$, $l = 1,\ldots,n_b$, where

$$\begin{aligned}n_a &= \max_{m=1,\ldots,n_y}(n_a^m), \\ n_b &= \max_{m=1,\ldots,n_y}\left(\max_{n=1,\ldots,n_u}(n_a^{m,n})\right).\end{aligned} \tag{38}$$

If the hyperbolic tangent is used as the nonlinear transfer function φ in the hidden layer of the neural model, one has

$$\frac{\mathrm{d}\varphi(z_i^m(\overline{x}_m(k)))}{\mathrm{d}z_i^m(\overline{x}_m(k))} = 1 - \tanh^2(z_i^m(\overline{x}_m(k))).$$

Let $a_l^m(k) = a_l^m(\overline{x}_m(k))$, $b_l^{m,n}(k) = b_l^{m,n}(\overline{x}_m(k))$. Redefining the variables $y_m(k) := y_m(k) - g_m(\overline{x}_m(k))$, $y_m(k-i) := y_m(k-i) - \overline{y}_m(k-i+1)$, $l = 1,\ldots,n_a$, $u_n(k-i) := u_n(k-i) - \overline{u}_n(k-i+1)$, $l = 1,\ldots,n_b$, the linear approximation of the nonlinear model (9), obtained at a time instant k, can be expressed as

$$\mathbf{A}(k, z^{-1})y(k) = \mathbf{B}(k, z^{-1})u(k), \tag{39}$$

where z^{-1} denotes the operator of a unit time delay, and $\mathbf{A}$ and $\mathbf{B}$ are given by Eqns. (40) and (41). The step-response coefficients comprising the dynamic matrix (28) are calculated from

$$s_j^{m,n}(k) = \sum_{i=1}^{\min(j,n_b)} b_i^{m,n}(k) - \sum_{i=1}^{\min(j-1,n_a)} a_i^m(k)s_{j-i}^{m,n}(k) \tag{42}$$

for all $m = 1,\ldots,n_y$, $n = 1,\ldots,n_u$, $j = 1,\ldots,N$.

$$\mathbf{A}(k,z^{-1})=\begin{bmatrix} 1+a_1^1(k)z^{-1}+\cdots+a_{n_a}^1(k)z^{-n_a} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1+a_1^{n_y}(k)z^{-1}+\cdots+a_{n_a}^{n_y}(k)z^{-n_a} \end{bmatrix}, \tag{40}$$

$$\mathbf{B}(k,z^{-1})=\begin{bmatrix} b_1^{1,1}(k)z^{-1}+\cdots+b_{n_b}^{1,1}(k)z^{-n_b} & \dots & b_1^{1,n_u}(k)z^{-1}+\cdots+b_{n_b}^{1,n_u}(k)z^{-n_b} \\ \vdots & \ddots & \vdots \\ b_1^{n_y,1}(k)z^{-1}+\cdots+b_{n_b}^{n_y,1}(k)z^{-n_b} & \dots & b_1^{n_y,n_u}(k)z^{-1}+\cdots+b_{n_b}^{n_y,n_u}(k)z^{-n_b} \end{bmatrix}. \tag{41}$$

$$\mathbf{G}(k)=\begin{bmatrix} \mathbf{S}_1(k,k+1) & \mathbf{0}_{n_y\times n_u} & \dots & \mathbf{0}_{n_y\times n_u} \\ \mathbf{S}_2(k,k+2) & \mathbf{S}_1(k,k+2) & \dots & \mathbf{0}_{n_y\times n_u} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_N(k,k+N) & \mathbf{S}_{N-1}(k,k+N) & \dots & \mathbf{S}_{N-N_u+1}(k,k+N) \end{bmatrix}. \tag{43}$$

5.2.2. Multi-Point Linearisation. In the single-point linearisation method, the linearisation is performed once for a given sampling instant k, and the same local linearised model is then used for the entire prediction horizon to determine the step-response. Although such a model can be very accurate for the current time instant k, its accuracy may deteriorate at the end of the prediction horizon. Conceptually, it would be better to perform linearisation N times for $k+1, k+2, \dots, k+N$ and obtain N independent local models. Next, these local linear models could be used for determining step-response coefficients. Let $\mathbf{S}_p(k,k+p)$ denote the step-response submatrix calculated at the current sampling instant k using the linearised model for the sampling instant $k+p$. The dynamic matrix, similarly to (28), is then given by Eqn. (43) For $k+p$, $p=1,\dots,N$ the linearisation point is

$$\begin{aligned}\overline{x}_m(k,k+p) &= \Big[\overline{u}_1(k-\tau^{m,1}+p)\dots\overline{u}_1(k-n_b^{m,1}+p)\dots \\ &\quad \overline{u}_{n_u}(k-\tau^{m,n_u}+p)\dots\overline{u}_{n_u}(k-n_b^{m,n_u}+p) \\ &\quad \overline{y}_m(k+p-1)\dots\overline{y}_m(k-n_a^m+p)\Big]^T.\end{aligned} \tag{44}$$

For $k+1$, the linearisation point is the same as in the single-point linearisation method (33). As p increases, the model is linearised taking into account the optimal input and output trajectories obtained at the previous sampling instant, i.e., $\Delta\mathbf{u}(k-1)$ and $\hat{\mathbf{y}}(k-1)$. More specifically, for linearisation purposes, $\overline{u}_n(k+p)=u_n(k+p|k-1)$ for $p\geq 0$ and $\overline{y}_m(k+p)=\hat{y}(k+p|k-1)$ for $p\geq 1$.

5.2.3. On-Line Calculation of the Nonlinear Free Trajectory. The nonlinear free trajectory $y_m^0(k+p|k)$, for $m=1,\dots,n_y$, $p=1,\dots,N$, is calculated recursively from the general prediction equation (19), where the output of the model is given by (21). Analogously to the MPC-NO algorithm, the "DMC-type" disturbance model (20) is also used. One has

$$y_m^0(k+p|k)=w_2^m(0)+\sum_{i=1}^{K^m}w_2^m(i)\varphi\Big(z_i^{m,0}(k+p|k)\Big)+d_m(k). \tag{45}$$

The quantities $z_i^{m,0}(k+p|k)$ are determined from (22) assuming no changes in control signals from a sampling instant k and replacing the output predictions by the corresponding values of the free trajectory, i.e., $u_n(k+p|k):=u_n(k-1)$ for $p\geq 0$, $\hat{y}_m(k+p|k):=y_m^0(k+p|k)$ for $p\geq 1$. One has

$$\begin{aligned} z_i^{m,0}(k+p|k) &= w_1^m(i,0) \\ &\quad+\sum_{n=1}^{n_u}\sum_{j=1}^{I_{uf}^{m,n}(p)}w_1^m(i,R^{m,n}+j)u_n(k-1) \\ &\quad+\sum_{n=1}^{n_u}\sum_{j=I_{uf}^{m,n}(p)+1}^{I_u^{m,n}}w_1^m(i,R^{m,n}+j) \\ &\quad\times u_n(k-\tau^{m,n}+1-j+p) \\ &\quad+\sum_{j=1}^{I_{yp}^m(p)}w_1^m(i,S^m+j)y_m^0(k-j+p|k) \\ &\quad+\sum_{j=I_{yp}^m(p)+1}^{n_a^m}w_1^m(i,S^m+j)y_m(k-j+p). \end{aligned} \tag{46}$$

6. Hybrid MPC Algorithm with Nonlinear Prediction, Linearisation and Nonlinear Optimisation (MPC-NPL-NO)

As was emphasised in Section 4, the computational burden of the MPC-NO algorithm depends on the initial point, $\mathbf{u}^0(k)$, of the nonlinear optimisation problem (13).

Because a gradient-based optimisation method of the SQP type is used, the algorithm may terminate in local minima. On the other hand, the suboptimal MPC-NPL algorithm solves only the quadratic optimisation problem (31), and hence the global solution to this task is always found. The idea of the hybrid MPC algorithm with Nonlinear Prediction, Linearisation and Nonlinear Optimisation (MPC-NPL-NO) is to find an initial point by means of the MPC-NPL algorithm and next solve the MPC-NO problem to refine the solution.

It is obvious that the second phase of the hybrid algorithm is necessary only if the superposition principle (27) is far from satisfaction. In other words, the MPC-NO nonlinear optimisation problem (13) is solved if for a given solution to the MPC-NPL optimisation problem (5.1), $\Delta\mathbf{u}^{\mathrm{NPL}}(k)$, the difference between the nonlinear prediction $\hat{\mathbf{y}}^{nl}(k)$ computed by means of the neural model and the linearised one is significant, i.e.,

$$\left\|\hat{\mathbf{y}}^{nl}(k) - \mathbf{G}(k)\Delta\mathbf{u}^{NPL}(k) - \mathbf{y}^{0}(k)\right\|^{2} > \varepsilon, \qquad (47)$$

where $\varepsilon > 0$ is adjusted by the user.

Thanks to the efficiency of the MPC-NPL algorithm, the initial solution determined in the first phase of the hybrid algorithm is usually close to the global minimum of the cost function $J(k)$ minimised in the second phase. When compared with the MPC-NO technique, the MPC-NPL-NO algorithm reduces the computational burden. Secondly, by finding a feasible initial point it is practically very unlikely that the MPC-NO algorithm will yield a local solution.

Naturally, the hybrid algorithm is suitable for significantly nonlinear processes, for which the performance of the MPC-NPL technique is not sufficient. One can also imagine that the second phase is executed at each sampling instant because full, nonlinear output constraints have to taken into account, which cannot be done in the quadratic programming problem solved in the MPC-NPL algorithm.

7. Simulation Results

7.1. Polymerisation Reactor Control System. The control process under consideration is a polymerization reaction taking place in a jacketed continuous stirred tank reactor depicted in Fig. 4 (Maner *et al.*, 1996). The reaction is the free-radical polymerization of methyl methacrylate with azo-bis-isobutyronitrile as an initiator and toluene as a solvent. The output $NAMW$ (Number Average Molecular Weight) is controlled by manipulating the inlet initiator flow rate F_I. The monomer flow rate F is the disturbance whose value is assumed constant. The polymerisation reactor is frequently used as a benchmark process for comparing nonlinear control strategies.

Four models of the process are used. The fundamental model (Maner *et al.*, 1996) is used as the real process

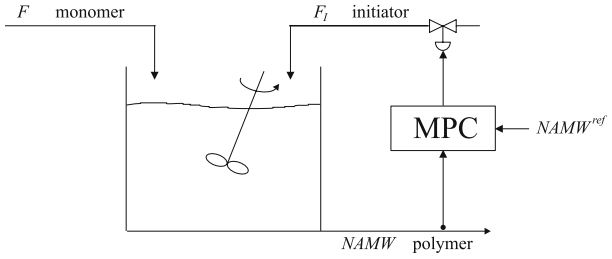

Fig. 4. Polymerisation reactor control system structure.

during simulations. An identification procedure is carried out. As a result, two local linear models for a low and a high $NAMW$ level and a neural one are obtained. All three empirical models have the same input arguments determined by $\tau = 2$, $n_a = n_b = 2$. The empirical models used in MPC algorithms are:

(a) a linear model for a low $NAMW$ level ($NAMW = 20000$)

$$y(k) = b_2^{\mathrm{low}}u(k-2) - a_1^{\mathrm{low}}y(k-1) - a_2^{\mathrm{low}}y(k-2), \qquad (48)$$

(b) a linear model for a high $NAMW$ level ($NAMW = 40000$)

$$y(k) = b_2^{\mathrm{high}}u(k-2) - a_1^{\mathrm{high}}y(k-1) - a_2^{\mathrm{high}}y(k-2), \qquad (49)$$

(c) a neural model containing six neurons in the hidden layer,

$$y(k) = g(u(k-2), y(k-1), y(k-2)), \qquad (50)$$

where $u = F_I$, $y = NAMW$.

The compared MPC strategies are:

(a) the linear MPC algorithm with the linear model for the low $NAMW$ level,

(b) the linear MPC algorithm with the linear model for the high $NAMW$ level,

(c) the nonlinear MPC-NO algorithm with the neural model,

(d) the nonlinear suboptimal MPC-NPL algorithm with the neural model,

(e) the nonlinear hybrid MPC-NPL-NO algorithm with the neural model.

The horizons are $N = 10$, $N_u = 3$, the weighting matrices $\mathbf{M}_p = 1$ and $\mathbf{\Lambda}_p = 0.5$. The manipulated variable is constrained, $F_{I\,\min} = 0.003$, $F_{I\,\max} = 0.06$, the sampling time is 1.8 min.

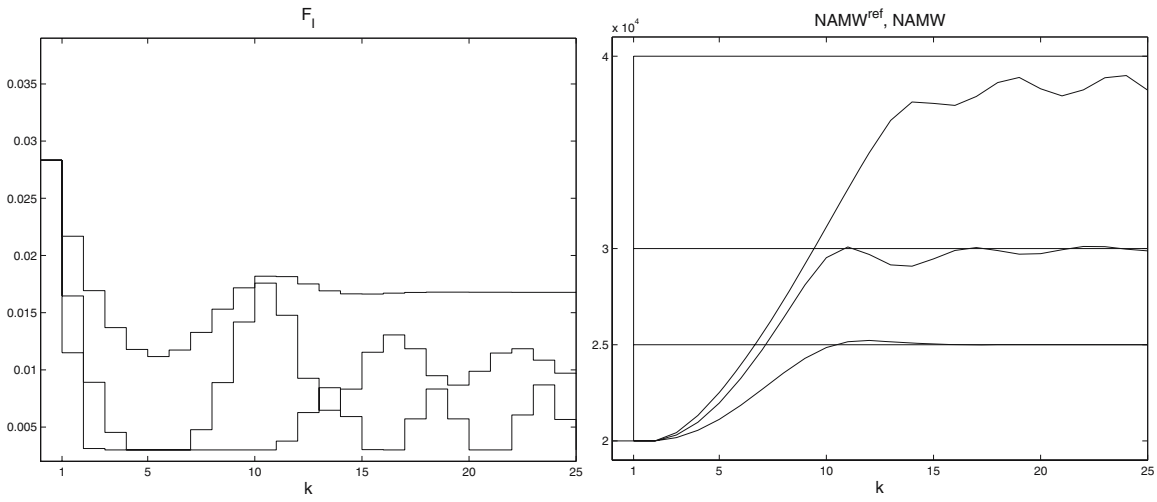

Fig. 5. Simulation results of the polymerisation reactor and the MPC algorithm and the linear model valid for a low $NAMW$ level: the set-point point changes from $NAMW = 20000$.

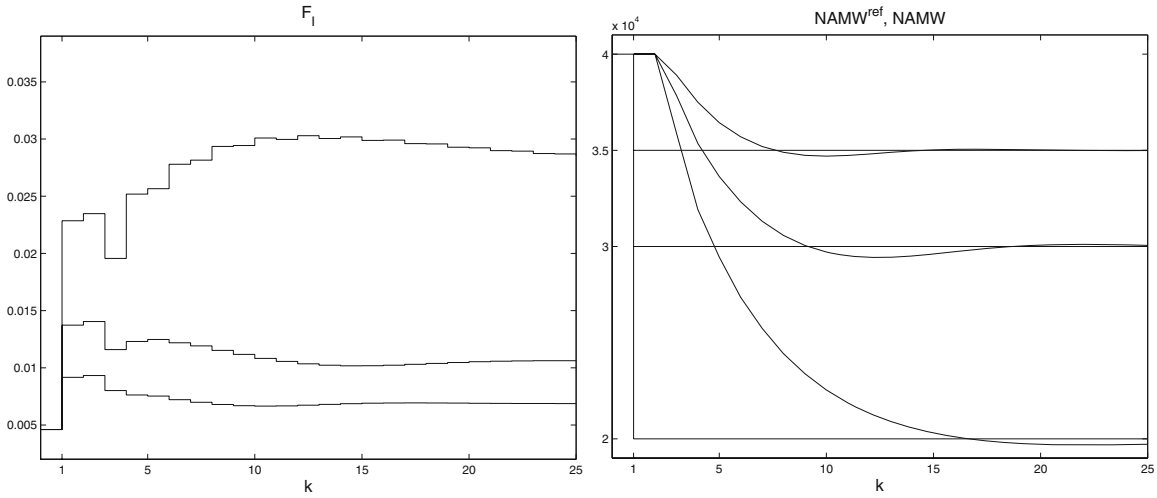

Fig. 6. Simulation results of the polymerisation reactor with the MPC algorithm with the linear model valid for a high $NAMW$ level: the set-point point changes from $NAMW = 40000$.

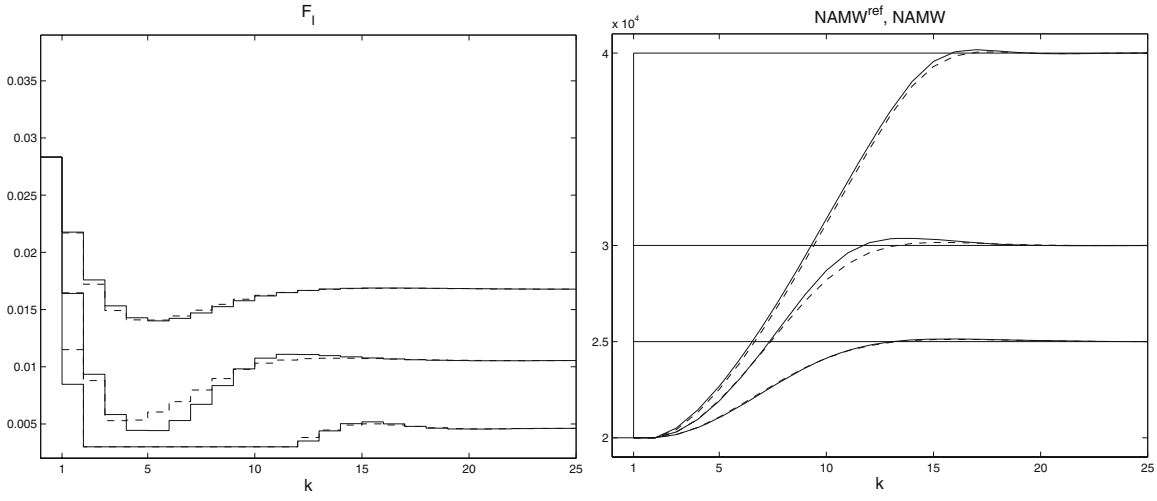

Fig. 7. Simulation results of the polymerisation reactor with the MPC-NPL (dashed line) and MPC-NO (solid line) algorithms with the neural network model: the set-point point changes from $NAMW = 20000$.

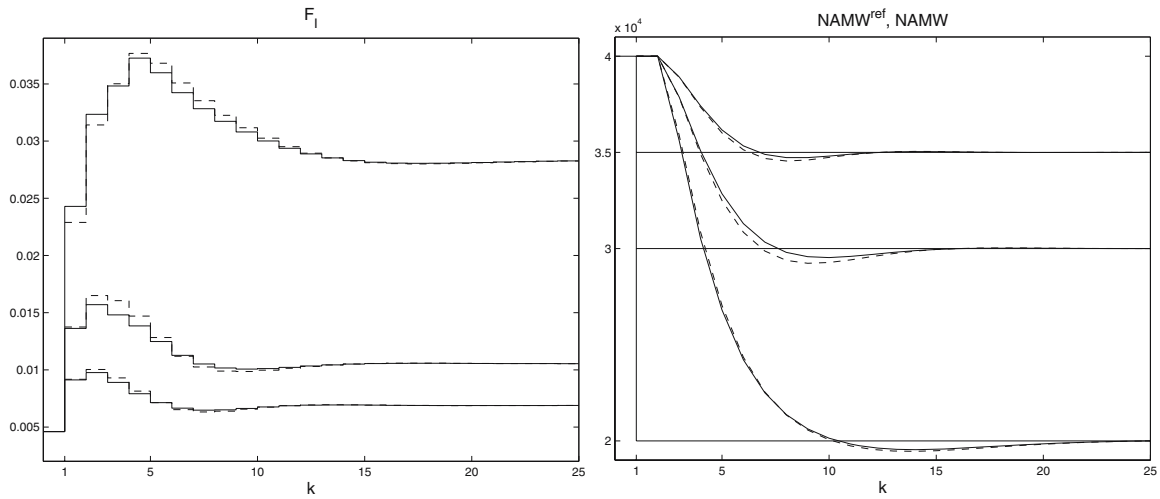

Fig. 8. Simulation results of the polymerisation reactor with the MPC-NPL (dashed line) and MPC-NO (solid line) algorithms with the neural network model: the set-point point changes from $NAMW = 40000$.

The first linear model is valid for the the low $NAMW$ level, and the resulting control algorithm works well for the smallest set-point change but exhibits unacceptable oscillatory behaviour for medium and big set-point changes, as is shown in Fig. 5. Analogously, the second linear model captures the process properties for the high $NAMW$ level, and the closed-loop response is fast enough for the smallest set-point change but very slow for bigger ones, as shown in Fig. 6. Simulation results of the MPC-NPL (using the single-point linearisation method) and MPC-NO algorithms with the same neural network model and for the set-point changes from $NAMW = 20000$ are depicted in Fig. 7, and those for the set-point changes from $NAMW = 40000$ are depicted in Fig. 8. In each case the closed-loop performance obtained in the suboptimal MPC-NPL algorithm is very close to that obtained in the computationally prohibitive MPC-NO approach. In the case of the polymerisation reactor, the single-point linearisation method gives very good results. The improvement resulting from using the multi-point linearisation is not significant, and thus it is not shown. The hybrid algorithm gives the same results as the MPC-NO algorithm.

7.2. Methanol-Water Distillation Column Control System. The plant under consideration is a methanol-water distillation column the structure of which is shown in Fig. 9 (Ławryńczuk, 2003). The distillation column is used to purify the input stream so that the top product is methanol while the bottom product contains only small quantities of alcohol. The composition of the top product is denoted by x_b, and the composition of the bottom product by x_b.

The plant has two manipulated variables: R – the reflux stream flow rate, and V – the vapour stream flow rate. Two fast single-loop PID controllers (denoted by LC) are used to stabilise the levels in the reflux tank and the bottom product tank. Two additional PID controllers (denoted by FC) are also used to control the actual streams of R and V. All the PID controllers comprise the basic control layer. In order to stabilise the compositions x_d and x_b of top and bottom products, a supervisory MPC algorithm is used. It treats the column as a two-input (R, V) two-output (x_d, x_b) process. The sampling time of this algorithm is 1 min. At the nominal operating point we have $x_{d0} = 0.95$, $x_{b0} = 0.05$, $R_0 = 33.3634$ kmol/h, $V_0 = 83.3636$ kmol/h. The compositions are expressed in molar fractions.

Three models of the process are used. Analogously to the polymerisation reactor case, the fundamental model (Ławryńczuk, 2003) is used as the real process during simulations. An identification procedure is carried out.

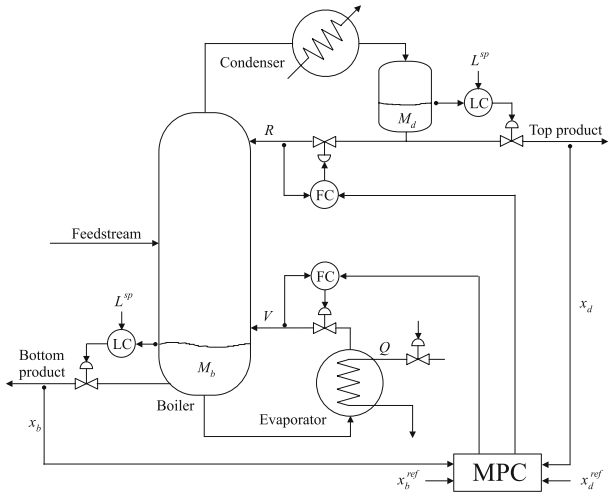

Fig. 9. Distillation column control system structure.

As a result, a linear model for the nominal operating point and a neural one are obtained. The empirical models have the same input arguments determined by $\tau^{m,n} = 1$, $n_a^m = n_b^{m,n} = 2$, $m = 1, 2$, $n = 1, 2$. The empirical models used in MPC algorithms are

(a) a linear model for the nominal operating point,

$$\begin{aligned} y_1(k) = {} & b_1^{1,1}u_1(k-1)+b_2^{1,1}u_1(k-2) \\ & + b_1^{1,2}u_2(k-1)+b_2^{1,2}u_2(k-2) \\ & - a_1^1 y_1(k-1)-a_2^1 y_1(k-2), \end{aligned} \quad (51)$$

$$\begin{aligned} y_2(k) = {} & b_1^{2,1}u_1(k-1)+b_2^{2,1}u_1(k-2) \\ & + b_1^{2,2}u_2(k-1)+b_2^{2,2}u_2(k-2) \\ & - a_1^2 y_1(k-1)-a_2^2 y_1(k-2), \end{aligned} \quad (52)$$

(b) a neural model comprised of two neural networks, each of which contains seven neurons in the hidden layer,

$$\begin{aligned} y_1(k) = g_1\Big(& u_1(k-1), u_1(k-2), u_2(k-1), \\ & u_2(k-2), y_1(k-1), y_1(k-2)\Big), \end{aligned} \quad (53)$$

$$\begin{aligned} y_2(k) = g_2\Big(& u_1(k-1), u_1(k-2), u_2(k-1), \\ & u_2(k-2), y_1(k-1), y_1(k-2)\Big), \end{aligned} \quad (54)$$

where $u_1 = R$, $u_2 = V$, $y_1 = x_d$, $y_2 = x_b$.

The compared MPC strategies are:

(a) the linear MPC algorithm with the linear model for the nominal operating point,

(b) the nonlinear MPC-NO algorithm with the neural model,

(c) the nonlinear suboptimal MPC-NPL algorithm with the neural model,

(d) the nonlinear hybrid MPC-NPL-NO algorithm with the neural model.

The horizons are $N = 10$, $N_u = 3$, the weighting matrices $\mathbf{M}_p$=diag$(5, 0.5)$ and $\mathbf{\Lambda}_p = \text{diag}(1.5, 1.5)$. The following constraints are imposed on the manipulated variables: $R_{\min} = R_0 - 20$ kmol/h, $R_{\max} = R_0 + 20$ kmol/h, $V_{\min} = V_0 - 20$ kmol/h, $V_{\max} = V_0 + 20$ kmol/h.

Simulation results of the MPC algorithm with the linear model are depicted in Fig. 10. Simulation results of the MPC-NPL (using the single-point linearisation method) and MPC-NO algorithms with the same neural network model are shown in Fig. 11. A few observations can be made. Both of the nonlinear algorithms with the neural network model work faster than the linear one, and the interactions between the top and bottom parts of the process are reduced. The differences between linear and nonlinear algorithms are clearly visible not only in the output, but also in input profiles, i.e., the manipulated variables change much faster in nonlinear algorithms. The performances of the MPC-NO and MPC-NPL algorithms are practically identical. Furthermore, as was in the case of the polymerisation reactor, the single-point linearisation method gives good results. Unlike the polymerisation reactor for which the linear MPC results in a poor performance (unstable or slow behaviour), this technique works satisfactorily for the distillation column, although the process is significantly nonlinear. This is because the set-points (compositions) change only moderately. Nevertheless, if the production scale is big, the advantages of nonlinear MPC over linear one are evident. Especially, nonlinear MPC is worth applying to the distillation process when on-line economic optimisation is used to adjust the operating point to the changes in the composition and flow rate of the feedstream (Tatjewski, 2007).

8. Conclusions

Because the nature of many industrial processes is nonlinear, the application of MPC algorithms with linear models may give poor closed-loop performance, e.g., instability. Nonlinear MPC algorithms with neural network models presented in this paper exhibit superior control in comparison with linear MPC techniques. Feedforward neural networks are used as process models. Having excellent approximation abilities, in comparison with popular fuzzy models they do not suffer from the "curse of dimensionality", which is troublesome in multivariable cases. Furthermore, unlike many fundamental models (e.g., distillation columns), feedforward neural models have a simple, regular structure. Hence, they can be easily incorporated into the described MPC algorithms and efficiently used on-line.

The emphasis is put on reliability, computational efficiency and closed-loop accuracy of the MPC algorithms considered. The MPC-NO algorithm, although potentially very accurate, has limited applicability, since nonlinear optimisation is used on-line. On the contrary, the MPC-NPL algorithm uses on-line only a quadratic programming procedure, and thus the necessity for full nonlinear optimisation is avoided. The hybrid algorithm combines advantages of both classes. As far as closed-loop accuracy is concerned, in the case of the presented examples the suboptimal MPC-NPL algorithm practically gives performance comparable to that obtained in the MPC-NO scheme. When the process is significantly nonlinear and linearisation-based algorithms are not sufficient or the nonlinear output constraints have to be satisfied, the hybrid algorithm is recommended. Although for the processes considered the single-point linearisation method gives good results, future research will em-

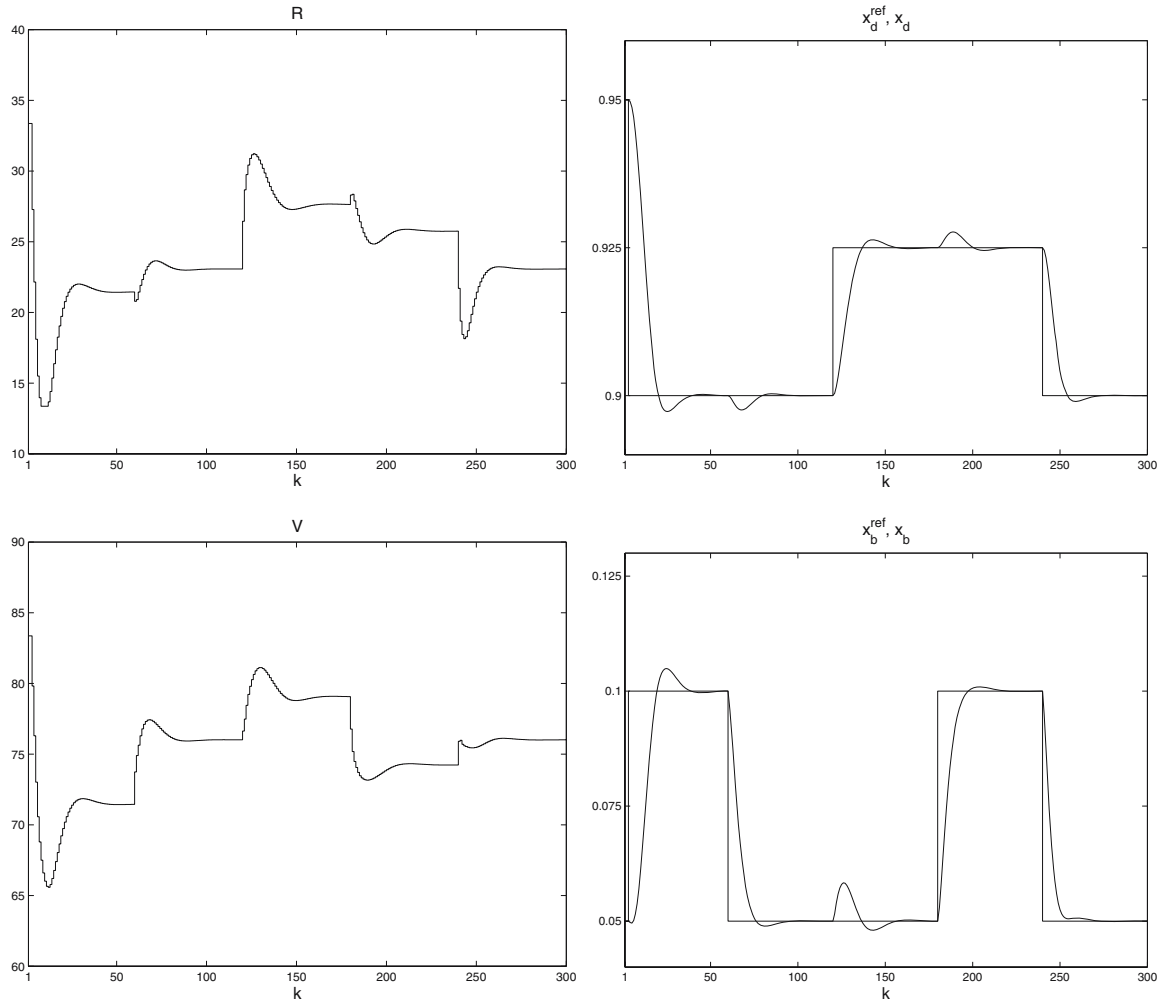

Fig. 10. Simulation results of the distillation column with the MPC algorithm with the linear model

brace the investigation of different linearisation methods and the application of the discussed algorithms to various processes.

The stability of the presented MPC algorithms with neural models can be practically achieved by proper tuning of the weighting matrices $\mathbf{M}_p$ and $\mathbf{\Lambda}_p$ in the cost function $J(k)$. Furthermore, all discussed algorithm classes can be combined with the stabilising dual-mode approach (Ławryńczuk and Tatjewski, 2004; Ławryńczuk, 2003) developed by Michalska and Mayne (1993). In this approach merely feasibility, rather than optimality, is sufficient to guarantee stability.

Acknowledgment

The work presented in this paper was supported by the Polish national budget funds for science in the years 2005–2007 as a research project.

References

Kesson B. M. and Toivonen H. T. (2006): *A neural network model predictive controller*. — J. Process Contr., Vol. 16, No. 3, pp. 937–946.

Bacic M., Cannon M. and Kouvaritakis B. (2002): *Feedback linearization MPC for discrete-time bilinear systems*. — Proc. 15-th *IFAC World Congress*, Barcelona, Spain, CD-ROM, paper 2391.

Babuška R., Sousa J. M. and Verbruggen H. B. (1999): *Predictive control of nonlinear systems based on fuzzy and neural models*. — Proc. *European Control Conf.*, Karlsruhe, Germany, CD-ROM, paper F1032-5.

Bazaraa M. S., Sherali J. and Shetty K. (1993): *Nonlinear Programming: Theory and Algorithms.* — New York: Wiley.

Bloemen H.H.J., van den Boom T. J. J. and Verbruggen H. B. (2001): *Model-based predictive control for Hammerstein-Wiener systems*. — Int. J. Contr., Vol. 74, No. 5, pp. 482–495.

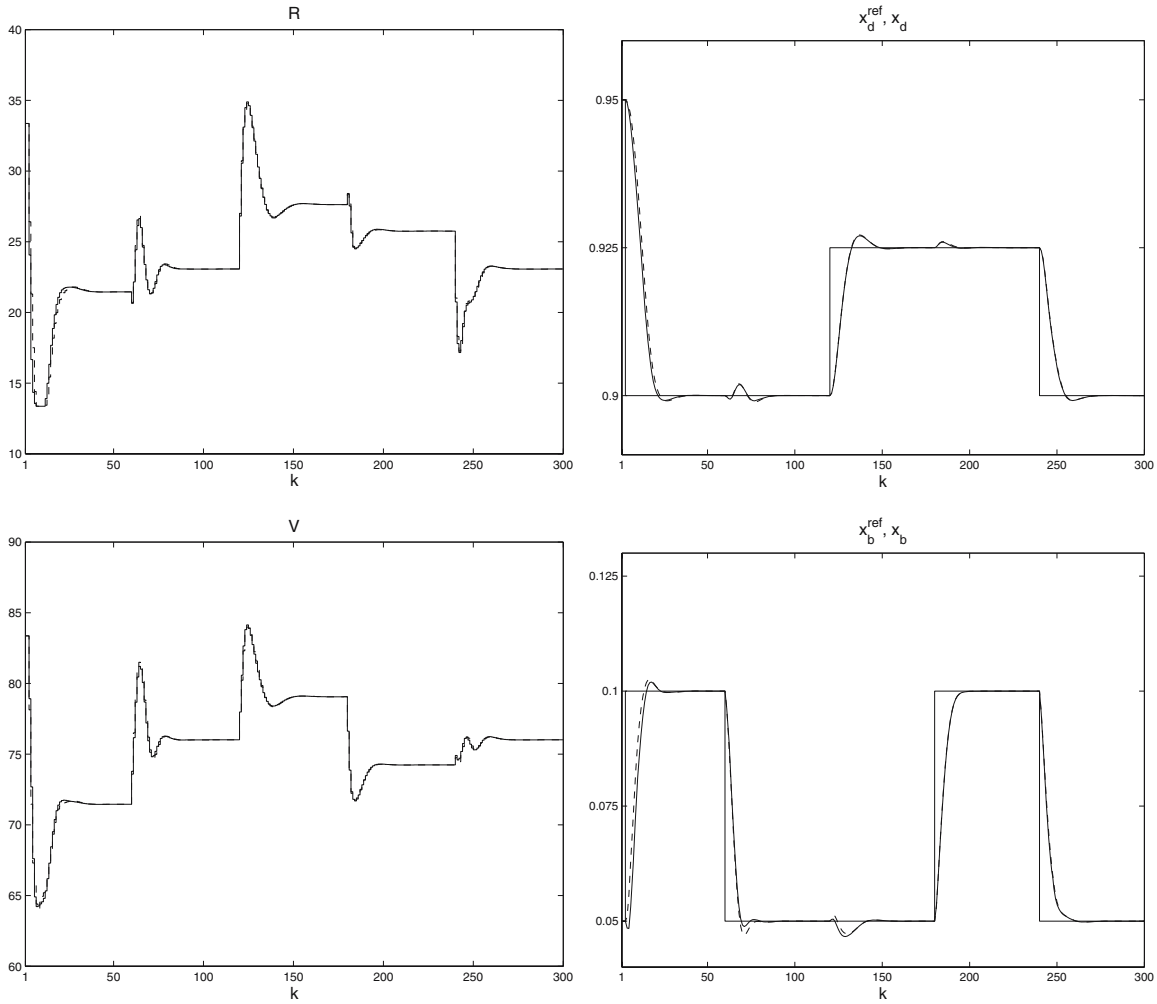

Fig. 11. Simulation results of the distillation column with the MPC-NPL (dashed line) and MPC-NO (solid line) algorithms with a neural network model

Brdyś M.A. and Tatjewski P. (2005): *Iterative algorithms for multilayer optimizing control*. — London: Imperial College Press/World Scientific.

Cavagnari L., Magni L. and Scattolini R. (1999): *Neural network implementation of nonlinear receding-horizon control*. — Neural Comput. Applic., Vol. 8, No. 1, pp. 86–92.

Clarke D. W., Mohtadi C. and Tuffs P. S. (1987): *Generalized predictive control – I. The basic algorithm.* — Automatica, Vol. 23, No. 2, pp. 137–148.

Cutler R. and Ramaker B. (1979): *Dynamic matrix control – A computer control algorithm.* — Proc. *AIChE National Meeting*, Houston.

Dutka A. and Ordys A. W. (2004): *The optimal non-linear generalised predictive control by the time-varying approximation*. — Proc. 10-th IEEE Int. Conf. *Methods and Models in Automation and Robotics*, Międzyzdroje, Poland, pp. 299–303.

Grimble M.J. and Ordys A.W. (2001): *Nonlinear predictive control for manufacturing and robotic applications*. — Proc. 7-th IEEE Int. Conf. *Methods and Models in Automation and Robotics*, Międzyzdroje, Poland, pp. 579–592.

Haykin S. (1999): *Neural Networks – A Comprehensive Foundation.* — Englewood Cliffs, NJ: Prentice Hall.

Henson M. A. (1998): *Nonlinear model predictive control: Current status and future directions.* — Comput. Chemi. Engi., Vol. 23, No. 2, pp. 187–202.

Hornik K., Stinchcombe M. and White H. (1989): *Multilayer feedforward networks are universal approximators*. — Neural Netw., Vol. 2, No. 5, pp. 359-366.

Hussain M. A. (1999): *Review of the applications of neural networks in chemical process control – Simulation and online implementation*. — Artifi. Intelli. Eng., Vol. 13, No. 1, pp. 55–68.

Kavsek B.K., Skrjanc I. and Matko D. (1997): *Fuzzy predictive control of a highly nonlinear pH process*. — Comput. Chem. Eng., Vol. 21, Supplement, pp. S613–S618.

Kouvaritakis B., Cannon M. and Rossiter J. A. (1999): *Nonlinear model based predictive control*. — Int. J. Contr., Vol. 72, No. 10, pp. 919–928.

Liu G. P., Kadirkamanathan V. and Billings S. A. (1998): *Predictive control for non-linear systems using neural networks*. — Int. J. Contr., Vol. 71, No. 6, pp. 1119–1132.

Ławryńczuk M. and Tatjewski P. (2006): *An efficient nonlinear predictive control algorithm with neural models and its application to a high-purity distillation process.* — Lecture Notes in Artificial Intelligence, Springer, Vol. 4029, pp. 76–85.

Ławryńczuk M. and Tatjewski P. (2004): *A stable dual-mode type nonlinear predictive control algorithm based on online linearisation and quadratic programming.* — Proc. 10-th IEEE Int. Conf. *Methods and Models in Automation and Robotics*, Międzyzdroje, Poland, pp. 503–510.

Ławryńczuk M. (2003): *Nonlinear model predictive control algorithms with neural models*. — Ph.D. thesis, Warsaw University of Technology, Warsaw, Poland.

Ławryńczuk M. and Tatjewski P. (2003): *An iterative nonlinear predictive control algorithm based on linearisation and neural models*. — Proc. *European Control Conf.*, Cambridge, U.K., CD-ROM, paper 339.

Ławryńczuk M. and Tatjewski P. (2002): *A computationally efficient nonlinear predictive control algorithm based on neural models*. — Proc. 8-th IEEE Int. Conf. *Methods and Models in Automation and Robotics*, Szczecin, Poland, pp. 781–786.

Ławryńczuk M. and Tatjewski P. (2001): *A multivariable neural predictive control algorithm*. — Proc. *IFAC Advanced Fuzzy-Neural Control Workshop*, Valencia, Spain, pp. 191–196.

Maciejowski J.M. (2002): *Predictive Control with Constraints.* — Harlow, U.K.: Prentice Hall.

Mahfouf M. and Linkens D.A. (1998): *Non-linear generalized predictive control (NLGPC) applied to muscle relaxant anaesthesia.* — Int. J. Contr., Vol. 71, No. 2, pp. 239–257.

Maner B.R., Doyle F.J., Ogunnaike B.A. and Pearson R.K. (1996): *Nonlinear model predictive control of a simulated multivariable polymerization reactor using second-order Volterra models.* — Automatica, Vol. 32, No. 9, pp. 1285–1301.

Michalska H. and Mayne D.Q. (1993): *Robust receding horizon control of constrained nonlinear systems*. — IEEE Trans. Automat. Cont., Vol. 38, No. 11, pp. 1623–1633.

Morari M. and Lee J. (1999): *Model predictive control: Past, present and future.* — Comput. Chem. Engi., Vol. 23, No. 4/5, pp. 667–682.

Nørgaard M., Ravn O., Poulsen N. K. and Hansen L. K. (2000): *Neural Networks for Modelling and Control of Dynamic Systems*. — London: Springer.

Osowski S. (1996): *Neural Networks — An Algorithmic Approach*. — Warsaw, Poland: WNT.

Parisini T., Sanguineti M. and Zoppoli R. (1998): *Nonlinear stabilization by receding-horizon neural regulators*. — Int. J. Contr., Vol. 70, No. 3, pp. 341–362.

Piche S., Sayyar-Rodsari B., Johnson D. and Gerules M. (2000): *Nonlinear model predictive control using neural networks*. — IEEE Contr. Syst. Mag., Vol. 20, No. 3, pp. 56–62.

Qin S. J. and Badgwell T. (2003): *A survey of industrial model predictive control technology.* — Contr. Eng. Pract., Vol. 11, No. 7, pp. 733–764.

Rossiter J. A. (2003): *Model-Based Predictive Control*. — Boca Raton, FL: CRC Press.

Sriniwas G. R. and Arkun Y.(1997): *A global solution to the non-linear model predictive control algorithms using polynomial ARX models*. — Comput. Chem. Engi., Vol. 21, No. 4, pp. 431–439.

Tatjewski P. (2007): *Advanced Control of Industrial Processes, Structures and Algorithms.* — London: Springer.

Tatjewski P. and Ławryńczuk M. (2006): *Soft computing in model-based predictive control*. — Int. J. Appl. Math. Comput. Sci., Vol. 16, No. 1, pp. 101–120.

Trajanoski Z. and Wach P. (1998): *Neural predictive control for insulin delivery using the subcutaneous route*. — IEEE Trans. Biomed. Eng., Vol. 45, No. 9, pp. 1122–1134.

Wang L. X. and Wan F. (2001): *Structured neural networks for constrained model predictive control*. — Automatica, Vol. 37, No. 8, pp. 1235–1243.

Yu D. L. and Gomm J. B. (2003): *Implementation of neural network predictive control to a multivariable chemical reactor*. — Contr. Eng. Pract., Vol. 11, No. 11, pp. 1315–1323.

Zheng A. (1997): *A computationally efficient nonlinear MPC algorithm*. — Proc. *American Control Conf.*, Albuquerque, pp. 1623–1627.

Received: 13 December 2006
Revised: 18 April 2007