Control and Cybernetics

vol. **37** (2008) No. 1

A computationally efficient stable dual-mode type nonlinear predictive control algorithm *†

by

Maciej Ławryńczuk[1] and Wojciech Tadej[2]

[1]Institute of Control and Computation Engineering
Faculty of Electronics and Information Technology
Warsaw University of Technology
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland

[2] Faculty of Mathematics and Natural Sciences
College of Sciences, Cardinal Stefan Wyszyński University
Warsaw, Poland

Abstract: This paper describes a computationally efficient (suboptimal) nonlinear predictive control algorithm. The algorithm uses a modified dual-mode approach which guarantees closed-loop stability. In order to reduce the computational burden, instead of online nonlinear optimisation used in the classical dual-mode control scheme, a nonlinear model of the plant is linearised on-line and a quadratic programming problem is solved. Calculation of the terminal set and implementation steps of the algorithm are detailed, especially for input-output models, which are widely used in practice.

Keywords: nonlinear model predictive control, dual-mode model predictive control, process control, linearisation, optimisation, quadratic programming, stability, constraints, terminal set.

1. Introduction

Model Predictive Control (MPC) is recognised as the only advanced control technique (i.e. more advanced than the well known PID approach) which has been very successful in practical applications and has influenced not only directions of development of industrial control systems but also research in this area (Henson, 1998; Maciejowski, 2002; Morari, 1999; Qin and Badgwell, 2003; Rossiter, 2003; Tatjewski, 2007). MPC algorithms have many advantages. In particular, they have a unique ability to take into account constraints imposed

*Submitted: October 2006; Accepted: February 2008

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

on both process inputs (manipulated variables) and outputs (controlled variables) or state variables, which usually decide on quality, economic efficiency and safety of production. Furthermore, MPC techniques are very efficient in multivariable process control. Various MPC approaches are nowadays used in numerous applications including chemicals, food processing, robotics, automotive and aerospace (Qin and Badgwell, 2003).

Because properties of many technological processes are nonlinear, different nonlinear MPC techniques have been developed (Henson, 1998; Morari and Lee, 1999; Tatjewski, 2007). The main difficulty is that a nonlinear model used for prediction entails the necessity of solving a nonlinear optimisation problem at each sampling instant on-line. Computational burden of such a problem is usually big and it may terminate at local minima. As a result, reliability of MPC with nonlinear optimisation is limited. In order to circumvent these difficulties various suboptimal MPC algorithms with on-line linearisation have been developed (Declercq and de Keyser, 1999; Garcia, 1984; Gattu and Zafiriou, 1992; Ławrynczuk, 2007; Ławrynczuk and Tatjewski, 2006; Lee and Ricker, 1994; Li and Biegler, 1989; Mollov et al., 2004; Mutha et al., 1997; Oliveira and Bieglerm 1995; Tatjewski and Ławryńczuk, 2006; Tatjewski, 2007). Suboptimal MPC algorithms require solving on-line only a quadratic programming problem. In practice, they give closed-loop control performance comparable to that obtained in MPC strategies with nonlinear optimisation repeated on-line at each sampling instant. As emphasised in Morari and Lee (1999), suboptimal MPC is the only method which is widely used in industry rather than MPC with nonlinear optimisation.

MPC algorithms have gained recognition and have been successfully applied in industrial practice for years before theoretical results concerning their stability analysis appeared. In the 1970s and 1980s stability was practically achieved by tuning horizon lengths and penalty factors. Such an approach received strong criticism as "playing games" (Bitmead et al., 1990). In the meantime, however, some stability criteria were obtained, but their applicability was limited to particular cases, especially, unconstrained linear MPC algorithms were mainly considered (Clarke and Mohtadi, 1989; Clarke et al., 1987; Rouhani and Mehra, 1982; Scattolini and Bittanti, 1990). Over the years several MPC algorithms with guaranteed stability in which constraints can be taken into account have been developed:

a) the algorithms with a terminal equality constraint (Bemporad, 1994; Chisci and Mosca, 1994; Clarke and Scattolini, 1991; Keerthi and Gilbert, 1988; Mayne and Michalska, 1990; Meadows et al., 1995; Scokaert et al., 1999; Scokaert and Clarke, 1994),

b) the algorithms with an infinite horizon (Meadows and Rawlings, 1993; Muske and Rawlings, 1993; Rawlings and Muske, 1993; Scokaert, 1997; Scokaert and Clarke, 1994),

c) the dual-mode algorithm (Chisci et al., 1996; Michalska and Mayne, 1993;

Scokaert et al., 1999),

d) the algorithms with a quasi-infinite horizon (Chen and Allgöwer, 1998; Magni et al., 2001; DeNicolao et al., 1998),

e) the algorithm with a contraction constraint (DeOliveira Kothare and Morari, 2000),

f) the algorithm with an artificial Lyapunov function (Bemporad, 2002).

Although suboptimal MPC algorithms with on-line linearisation and quadratic programming are widely used in practice, algorithms with guaranteed stability need on-line nonlinear optimisation. The gap between practice and theory is evident. Stability issues of suboptimal MPC are researched exceptionally infrequently (Ławrynczuk, 2004; Marusak and Tatjewski, 2003).

The additional terminal equality constraint, the purpose of which is to bring the state of the plant to the equilibrium point (for example to – the origin), guarantees stability, provided that at each sampling instant the optimisation problem is feasible and the global solution is found (Maciejowski, 2002; Mayne et al., 2000; Mayne and Michalska, 1990). Terminally constrained algorithms can be effectively developed and implemented on-line only when the model is linear, because the resulting optimisation problem is convex (Bemporad, 1994; Chisci and Mosca, 1994; Clarke and Scattolini, 1991; Scokaert and Clarke, 1994). When nonlinear MPC with the terminal equality constraint is considered, two major difficulties emerge. First of all, the optimisation problem is nonlinear, usually non-convex, it is practically impossible to find the global solution at each sampling instant in real-time. Secondly, the stabilising equality constraint poses a serious problem for the optimisation routine. To circumvent these difficulties, the dual-mode MPC scheme can be used, in which a terminal inequality constraint and an additional feedback local controller guarantee stability (Michalska and Mayne, 1993; Scokaert et al., 1999). In this approach merely feasibility, rather then optimality, is sufficient to guarantee stability. It means that it is only necessary to find a feasible solution to the MPC optimisation problem, this solution does not need to be the global or even a local minimum to guarantee stability of the control algorithm.

This paper presents a computationally efficient (suboptimal) dual-mode type MPC algorithm for nonlinear processes. It is based on the suboptimal MPC algorithm with Nonlinear Prediction and Linearisation (MPC-NPL) (Ławrynczuk, 2007; Tatjewski, 2007; Tatjewski and Ławrynczuk, 2006). At each sampling instant a nonlinear model of the process is linearised on-line and a free trajectory is calculated. The algorithm needs solving on-line only a quadratic programming problem. In order to guarantee closed-loop stability a modified dual-mode approach is used. In the classical dual-mode MPC a nonlinear stabilising constraint has to be taken into account, which means that the resulting optimisation problem is nonlinear. In order to formulate an easy to solve quadratic programming problem, this nonlinear constraint is transformed into a set of linear constraints. The contribution of the paper is twofold: *i)* it introduces

the stable dual-mode type predictive control algorithm with linearisation and discusses its implementation for input-output models that are widely used in practice, *ii)* it specifies the calculation of the terminal set, which can be also used in any dual-mode MPC algorithm, also when state-space models are used.

The organisation of the paper is as follows. First, in Section 2, the suboptimal MPC-NPL algorithm (MPC-NPL) and the dual-mode MPC algorithm are shortly presented. Section 3 describes the practical implementation of the suboptimal, stable dual-model type algorithm with on-line linearisation, quadratic programming and input-output models. Section 4 discusses calculation of the terminal set, which determines the stabilising constraint. Simulation results of the algorithm applied to a high-purity high-pressure ethylene-ethane distillation column are presented in Section 5, and the paper is summarised in Section 6.

2. Model Predictive Control algorithms

Although a number of different MPC techniques have been developed, the main idea (i.e. the explicit application of a process model, optimisation of a cost function and the receding horizon) is always the same (Maciejowski, 2002; Rossiter, 2003; Tatjewski, 2007). At each consecutive sampling instant k a set of future controls $\boldsymbol{u}(k)$ or corresponding increments $\Delta\boldsymbol{u}(k)$

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

is calculated. It is assumed that $\Delta u(k+p|k) = 0$ for $p \geq N_u$, where N_u is the control horizon. Usually, the objective is to minimise the differences between the reference trajectory $y^{ref}(k+p|k)$ and predicted values of the output $\hat{y}(k+p|k)$ over the prediction horizon N and to penalise excessive control increments. The following cost function is usually applied

$$J(k) = \sum_{p=1}^{N} \left\| y^{ref}(k+p|k) - \hat{y}(k+p|k) \right\|^2_{\boldsymbol{M}_p} + \sum_{p=0}^{N_u-1} \left\| \Delta u(k+p|k) \right\|^2_{\boldsymbol{\Lambda}_p}. \tag{2}$$

In general, a Multi-Input Multi-Output (MIMO) process is considered with n_u inputs (manipulated variables) and n_y outputs (controlled variables), i.e. $u(k) \in \Re^{n_u}$, $y(k) \in \Re^{n_y}$. $\boldsymbol{M}_p \geq 0$ and $\boldsymbol{\Lambda}_p > 0$ are weighting matrices of dimensionality $n_y \times n_y$ and $n_u \times n_u$, respectively. Typically, $N_u < N$, which decreases dimensionality of the optimisation problem. Only the first n_u elements of the determined sequence (1) are applied to the process, the control law is

$$u(k) = u(k|k) \qquad \text{or} \qquad u(k) = \Delta u(k|k) + u(k-1). \tag{3}$$

At the next sampling instant, $k+1$, the prediction is shifted one step forward and the whole procedure is repeated. In the simplest and the most common case,

the reference trajectory is not known in advance, hence $y^{ref}(k+p|k) = y^{sp}(k)$ for all $p = 1, \ldots, N$, where $y^{sp}(k)$ is the current set-point.

Considering constraints imposed on input and output variables, future control increments are determined from the following optimisation problem

$$\begin{aligned} &\min_{\Delta \boldsymbol{u}(k)} \{J(k)\} \\ &\text{subject to} \\ &\quad u^{\min} \le u(k+p|k) \le u^{\max}, \quad p = 0, \ldots, N_u - 1 \\ &\quad -\Delta u^{\max} \le \Delta u(k+p|k) \le \Delta u^{\max}, \quad p = 0, \ldots, N_u - 1 \\ &\quad y^{\min} \le \hat{y}(k+p|k) \le y^{\max}, \quad p = 1, \ldots, N. \end{aligned} \tag{4}$$

For prediction purposes, a dynamic model of the process is used. The model of a MIMO process is comprised of n_y Multi-Input Single-Output (MISO) models. Let each consecutive MISO model be described by the following nonlinear discrete-time equation

$$\begin{aligned} y_m(k) = f_m^{NARX}(&u_1(k-\tau^{m,1}), \ldots, u_1(k-n_B^{m,1}), \ldots, \\ &u_{n_u}(k-\tau^{m,n_u}), \ldots, u_{n_u}(k-n_B^{m,n_u}), \\ &y_m(k-1), \ldots, y_m(k-n_A^m)) \end{aligned} \tag{5}$$

where $f_m^{NARX} : \Re^{n_A^m + \sum_{n=1}^{n_u}(n_B^{m,n} - \tau^{m,n} + 1)} \longrightarrow \Re$, $m = 1, \ldots, n_y$. Integers $\tau^{m,n}$, $n_{n_A}^{m,n}$, $n_{n_B}^{m,n}$ where $m = 1, \ldots, n_y$, $n = 1, \ldots, n_u$ and $\tau^{m,n} \le n_{n_B}^{m,n}$, define the order of the dynamics. The model used is of Nonlinear Auto Regressive with eXternal input (NARX) class. The rudimentary suboptimal MPC-NPL algorithm and its stable version described in this paper do not restrict the structure of the model. It is only necessary that the functions f_m be differentiable in order to linearise the model on-line.

2.1. MPC algorithm with Nonlinear Prediction and Linearisation (MPC-NPL)

In general, two approaches to nonlinear MPC can be distinguished: MPC with Nonlinear Optimisation (MPC-NO) and suboptimal MPC. If for prediction purposes a nonlinear model (e.g. a neural network, a fuzzy system) is used without any simplifications, at each sampling instant a nonlinear optimisation problem (4) has to be solved on-line.

It is necessary to emphasise the fact that the difficulty of the MPC-NO optimisation problem (4) is twofold. First of all, it is nonlinear, computationally demanding, its computational burden is big. Secondly, it may be non-convex and even multi-modal. Unfortunately, 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 predefined time limit, as it is required in on-line control. Global optimisation

techniques substantially increase the computational burden, yet they still give no guarantee that the global solution is found. In this paper a suboptimal MPC algorithm, which needs solving on-line only a quadratic programming problem is recommended.

In the MPC-NPL algorithm at each sampling instant k the nonlinear model is used on-line twice: to find a local linearisation and a nonlinear free trajectory (Ławrynczuk, 2007; Ławrynczuk and Tatjewski, 2006; Tatjewski, 2007; Tatjewski and Ławrynczuk, 2006). It is assumed that the output prediction $\hat{\boldsymbol{y}}(k)$ can be expressed as the sum of a forced trajectory, which depends only on the future (on future input moves $\Delta\boldsymbol{u}(k)$) and a free trajectory $\boldsymbol{y}^0(k)$, which depends only on the past

$$\hat{\boldsymbol{y}}(k) = \boldsymbol{G}(k)\Delta\boldsymbol{u}(k) + \boldsymbol{y}^0(k) \tag{6}$$

where

$$\hat{\boldsymbol{y}}(k) = \begin{bmatrix} \hat{y}(k+1|k) \\ \vdots \\ \hat{y}(k+N|k) \end{bmatrix}, \quad \boldsymbol{y}^0(k) = \begin{bmatrix} y^0(k+1|k) \\ \vdots \\ y^0(k+N|k) \end{bmatrix} \tag{7}$$

are vectors of length n_yN and $\boldsymbol{G}(k)$ is a dynamic matrix of dimensionality $n_yN \times n_uN_u$. It is comprised of step-response coefficients of the linearised model

$$\boldsymbol{G}(k) = \begin{bmatrix} \boldsymbol{S}_1(k) & 0 & \dots & 0 \\ \boldsymbol{S}_2(k) & \boldsymbol{S}_1(k) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{S}_N(k) & \boldsymbol{S}_{N-1}(k) & \dots & \boldsymbol{S}_{N-N_u+1}(k) \end{bmatrix}. \tag{8}$$

For the considered MIMO process having n_u inputs and n_y outputs the step-response submatrices are

$$\boldsymbol{S}_j(k) = \begin{bmatrix} s_j^{1,1}(k) & \dots & s_j^{1,n_u}(k) \\ \vdots & \ddots & \vdots \\ s_j^{n_y,1}(k) & \dots & s_j^{n_y,n_u}(k) \end{bmatrix}. \tag{9}$$

Both the free trajectory and the dynamic matrix are calculated on-line from the nonlinear model taking into account the current state of the plant.

On the one hand, the suboptimal prediction calculated from (6) is different from the optimal one determined from the nonlinear model as it is done in the MPC-NO algorithm with nonlinear optimisation. On the other hand, thanks to using (6), the general optimisation problem (4) becomes the following quadratic

programming task

$$\begin{aligned}\min_{\Delta\boldsymbol{u}(k),\ \boldsymbol{\varepsilon}^{\min},\ \boldsymbol{\varepsilon}^{\max}} &\left\{ \left\|\boldsymbol{y}^{sp}(k) - \boldsymbol{G}(k)\Delta\boldsymbol{u}(k) - \boldsymbol{y}^{0}(k)\right\|_{\boldsymbol{M}}^{2} \right.\\ &\left. + \|\Delta\boldsymbol{u}(k)\|_{\boldsymbol{\Lambda}}^{2} + \rho^{\min}\left\|\boldsymbol{\varepsilon}^{\min}\right\|^{2} + \rho^{\max}\left\|\boldsymbol{\varepsilon}^{\max}\right\|^{2} \right\}\\ &\text{subject to}\\ &\boldsymbol{u}^{\min} \le \boldsymbol{J}\Delta\boldsymbol{u}(k) + \boldsymbol{u}^{k-1}(k) \le \boldsymbol{u}^{\max}\\ &-\Delta\boldsymbol{u}^{\max} \le \Delta\boldsymbol{u}(k) \le \Delta\boldsymbol{u}^{\max}\\ &\boldsymbol{y}^{\min} - \boldsymbol{\varepsilon}^{\min} \le \boldsymbol{y}^{0}(k) + \boldsymbol{G}(k)\Delta\boldsymbol{u}(k) \le \boldsymbol{y}^{\max} + \boldsymbol{\varepsilon}^{\max}\\ &\boldsymbol{\varepsilon}^{\min} \ge 0, \quad \boldsymbol{\varepsilon}^{\max} \ge 0\end{aligned} \tag{10}$$

where

$$\boldsymbol{y}^{sp}(k) = \begin{bmatrix} y^{sp}(k) \\ \vdots \\ y^{sp}(k) \end{bmatrix}, \quad \boldsymbol{y}^{\min} = \begin{bmatrix} \Delta y^{\min} \\ \vdots \\ \Delta y^{\min} \end{bmatrix}, \quad \boldsymbol{y}^{\max} = \begin{bmatrix} \Delta y^{\max} \\ \vdots \\ \Delta y^{\max} \end{bmatrix} \tag{11}$$

are vectors of length $n_y N$,

$$\boldsymbol{u}^{\min} = \begin{bmatrix} u^{\min} \\ \vdots \\ u^{\min} \end{bmatrix}, \quad \boldsymbol{u}^{\max} = \begin{bmatrix} u^{\max} \\ \vdots \\ u^{\max} \end{bmatrix}, \quad \Delta\boldsymbol{u}^{\max} = \begin{bmatrix} \Delta u^{\max} \\ \vdots \\ \Delta u^{\max} \end{bmatrix} \tag{12}$$

are vectors of length $n_u N_u$, $\boldsymbol{M} = diag(\boldsymbol{M}_1, \ldots, \boldsymbol{M}_N)$, $\boldsymbol{\Lambda} = diag(\boldsymbol{\Lambda}_0, \ldots, \boldsymbol{\Lambda}_{N_u-1})$ are weighting matrices of dimensionality $n_y N \times n_y N$ and $n_u N_u \times n_u N_u$, respectively. $\boldsymbol{J}$ is a matrix of dimensionality $n_u N_u \times n_u N_u$ and $\boldsymbol{u}^{k-1}(k)$ is a vector of length $n_u N_u$

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{I}_{n_u\times n_u} & \boldsymbol{0}_{n_u\times n_u} & \boldsymbol{0}_{n_u\times n_u} & \cdots & \boldsymbol{0}_{n_u\times n_u} \\ \boldsymbol{I}_{n_u\times n_u} & \boldsymbol{I}_{n_u\times n_u} & \boldsymbol{0}_{n_u\times n_u} & \cdots & \boldsymbol{0}_{n_u\times n_u} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{I}_{n_u\times n_u} & \boldsymbol{I}_{n_u\times n_u} & \boldsymbol{I}_{n_u\times n_u} & \cdots & \boldsymbol{I}_{n_u\times n_u} \end{bmatrix}, \quad \boldsymbol{u}^{k-1}(k) = \begin{bmatrix} u(k-1) \\ \vdots \\ u(k-1) \end{bmatrix} \tag{13}$$

where $\boldsymbol{I}$ and $\boldsymbol{0}$ are identity and zero matrices of appropriate dimensionality.

If output constraints have to be taken into account, the MPC optimisation task may be affected by the infeasibility problem. In order to cope with such a situation, the output constraints have to be softened by slack variables (Maciejowski, 2002; Tatjewski, 2007). A quadratic penalty for constraint violations is used in the MPC optimisation problem (10), $\boldsymbol{\varepsilon}^{\min}$ and $\boldsymbol{\varepsilon}^{\max}$ are vectors of length N comprising the slack variables and $\rho^{\min}$, $\rho^{\max} > 0$ are weights.

The structure of the MPC-NPL algorithm is depicted in Fig. 1. At each sampling instant k the following steps are repeated:

1. Linearisation of the nonlinear model: obtain the matrix $\boldsymbol{G}(k)$.

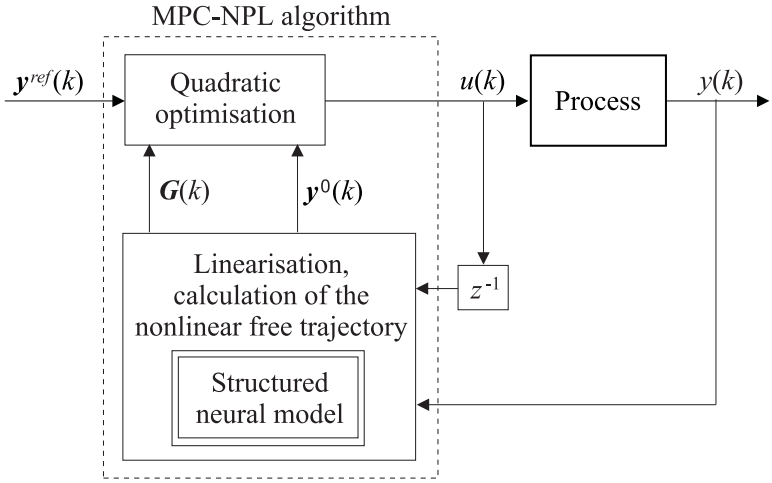

Figure 1. Structure of the MPC algorithm with Nonlinear Prediction and Linearisation (MPC-NPL)

2. Find the nonlinear free trajectory $\boldsymbol{y}^0(k)$ using the nonlinear model.
3. Solve the quadratic programming problem (10) to determine $\Delta\boldsymbol{u}(k)$.
4. Apply $u(k) = \Delta u(k|k) + u(k-1)$.
5. Set $k := k+1$, go to step 1.

The nonlinear model is used on-line twice at each sampling instant. At first, it is linearised, from the linearisation the step-response coefficients of the linearised model and the dynamic matrix are obtained. Next, the nonlinear free trajectory is calculated taking into account only the influence of the past. A detailed description of the MPC-NPL algorithm, in particular for neural network and fuzzy neural network models can be found in Ławryńczuk (2007), Ławryńczuk and Tatjewski (2006), Tatjewski (2007), Tatjewski and Ławryńczuk (2006).

2.2. Stable dual-mode MPC algorithm

In stable dual-mode MPC (Michalska and Mayne, 1993; Scokaert et al., 1999) a nonlinear state-space model of the plant is used

$$x(k+1) = f(x(k), u(k)) \tag{14}$$

where $x(k) \in \Re^{n_x}$, $u(k) \in \Re^{n_u}$, $f : \Re^{n_x+n_u} \to \Re^{n_x}$ is differentiable in the origin, $f(0,0) = 0$. A perfect model and a disturbance-free case are assumed. For the sake of simplicity of theoretical analysis the so called "regulator problem" is considered, i.e. the objective of the control algorithm is to drive the state of the

process to zero. The equilibrium point is equivalent to the origin with respect to the practical implementation of stable MPC-NPL algorithm (Section 3) this assumption is relaxed.

In MPC algorithms with linear models the stabilising equality terminal constraint

$$x(k+N|k) = 0 \tag{15}$$

can be effectively used to reduce the number of decision variables, for example as it is done in Constrained Receding Horizon Predictive Control (CRHPC) scheme (Clarke and Scattolini, 1991). In nonlinear MPC such a constraint significantly increases the complexity of the nonlinear optimisation problem. A straightforward idea used in the dual-mode algorithm (Michalska and Mayne, 1993; Scokaert et al., 1999) is to employ, instead of the equality constraint (15), the stabilising inequality terminal constraint

$$x(k+N|k) \in \Omega_\alpha \tag{16}$$

where Ω_α is a convex neighbourhood of the origin. The terminal set Ω_α is defined as

$$\Omega_\alpha = \left\{ x(k) \in \Re^{n_x} : \|x(k)\|^2_{\boldsymbol{P}} \leq \alpha \right\} \tag{17}$$

where the matrix $\boldsymbol{P} > 0$ (see Section 4).

According to the dual-model approach, values of manipulated variables are calculated in two different ways, the method used depends on the current state of the process. Outside the terminal set Ω_α a nonlinear MPC controller calculates the controls whereas an additional, usually linear, controller is used inside this set. The objective of the additional feedback controller is to bring the state of the process to the origin. The control law is

$$u(k) = \begin{cases} u(k|k) & \text{if} \quad x(k) \notin \Omega_\alpha \\ \boldsymbol{K}x(k) & \text{if} \quad x(k) \in \Omega_\alpha \end{cases} . \tag{18}$$

The performance index, whose minimisation leads to calculation of future controls in the MPC scheme used in the dual-mode algorithm when $x(k) \notin \Omega_\alpha$, is

$$J(k) = \sum_{p=0}^{N-1} \theta(x(k+p|k)) \left(\|x(k+p|k)\|^2_{\boldsymbol{Q}} + \|u(k+p|k)\|^2_{\boldsymbol{R}} \right) \tag{19}$$

where

$$\theta(x(k+p|k)) = \begin{cases} 1 & \text{if} \quad x(k+p|k) \notin \Omega_\alpha \\ 0 & \text{if} \quad x(k+p|k) \in \Omega_\alpha \end{cases} \tag{20}$$

$\boldsymbol{Q}$, $\boldsymbol{R} > 0$ and $x(k|k) = x(k)$. In general, constraints can be imposed both on state an input vectors

$$\begin{aligned} x(k+p|k) &\in X, \quad p = 1, \ldots, N-1 \\ u(k+p|k) &\in U, \quad p = 0, \ldots, N_u - 1 \end{aligned} \tag{21}$$

where the sets X and U are closed and $(0,0) \in interior(X \times U)$. Future values of manipulated variables, $\boldsymbol{u}(k)$, are found in the dual-mode MPC scheme when $x(k) \notin \Omega_\alpha$ from the nonlinear optimisation problem

$$\begin{aligned} &\max_{\boldsymbol{u}(k)} J(k) \\ &\text{subject to} \\ &\quad x(k+p|k) \in X, \quad p = 1, \ldots, N-1 \\ &\quad u(k+p|k) \in U, \quad p = 0, \ldots, N_u - 1 \\ &\quad x(k+N|k) \in \Omega_\alpha. \end{aligned} \tag{22}$$

The nonlinear MPC controller, which is used if the current state of the plant does not belong to the terminal set Ω_α, takes into account the constraints (21) explicitly in the optimisation problem, whereas the linear control law $u(k) = \boldsymbol{K}x(k)$, which is used if the state belongs to the set Ω_α, despite being unconstrained, must never violate the constraints. In fact, constraints are taken into account during calculating off-line the terminal set Ω_α as thoroughly discussed in Section 4.

The suboptimal dual-mode MPC algorithm can be summarised as follows

1. Set $\mu \in (0, 1]$.
2. For $k = 0$: if $x(0) \in \Omega_\alpha$ apply $u(0) = \boldsymbol{K}x(0)$. Otherwise, calculate a feasible solution $\boldsymbol{u}(0)$ to the nonlinear optimisation problem (22) and corresponding state trajectory $\boldsymbol{x}(0)$. Apply $u(0) = u(0|0)$.
3. For $k \geq 1$: if $x(k) \in \Omega_\alpha$ apply $u(k) = \boldsymbol{K}x(k)$. Otherwise, calculate a feasible solution $\boldsymbol{u}(k)$ to the nonlinear optimisation problem (22) and corresponding state trajectory $\boldsymbol{x}(k)$ which satisfy the condition

$$J(k) \leq J(k-1) - \mu \left(\|x(k-1)\|^2_{\boldsymbol{Q}} + \|u(k-1)\|^2_{\boldsymbol{R}} \right) \tag{23}$$

using

$$\boldsymbol{u}(k) = \begin{bmatrix} u(k|k-1) \\ \cdots \\ u(k+N-2|k-1) \\ \boldsymbol{K}x(k+N-1|k-1) \end{bmatrix} \tag{24}$$

as an initial guess, where $u(k+p|k-1) = u(k+N_u-2|k-1)$ for $p \geq N_u$. Apply $u(k) = u(k|k)$.

The main advantage of the algorithm is its suboptimality, i.e. feasibility of the nonlinear optimisation problem (22) and the property (23) of the performance index are sufficient to guarantee stability. In other words, the determined control sequence $\boldsymbol{u}(k)$ must be always feasible, but it does not need to be the global or even a local minimum of the optimisation problem (22). The initial state, $x(0)$, must belong to the region, where the optimisation problem is feasible. Stability proof of the discrete-time version of the algorithm can be found

in Scokaert et al. (1999), it corresponds to the original work by H. Michalska and D. Q. Mayne (Michalska and Mayne, 1993). In short, to prove stability it is sufficient to show that each trajectory of the process reaches the terminal set Ω_α within finite time, whereas stability in this set is guaranteed by the stabilising control law $u(k) = \boldsymbol{K}x(k)$.

Using the dual-mode approach, stabilising modifications can be introduced into the nonlinear Dynamic Matrix Control (DMC) algorithm which uses a step-response type fuzzy model (Marusak and Tatjewski, 2003). In this work a more general approach is presented. The described algorithm uses input-output NARX models, for example neural networks of various kinds. The algorithm does not restrict the structure of the model.

3. Stable MPC algorithm with Nonlinear Prediction and Linearisation (MPC-NPL)

From the practical point of view, computational efficiency is the most attractive feature of any nonlinear MPC algorithm. It should be considered in two aspects. Firstly, the algorithm should be suboptimal, the necessity of finding at each sampling instant the global solution to the nonlinear optimisation problem should be relaxed. Secondly, stabilising modifications to the standard MPC optimisation problem should not lead to increasing the complexity of the optimisation problem. The dual-mode MPC algorithm discussed in Section 2 has these properties: feasibility of the nonlinear optimisation problem, rather than optimality, implies stability, whereas the additional inequality constraint (16) is much easier to satisfy than the equality one (15). Furthermore, in the stable MPC-NPL algorithm, described in this paper, to reduce the computational burden, instead of non-convex nonlinear optimisation used in the classical dual-model MPC, quadratic programming is used. The main difficulty results from the necessity of satisfying the stabilising nonlinear inequality constraint (16). In order to make it possible to use the quadratic programming approach, this constraint is transformed into a set of linear inequality constraints.

Although the algorithm presented in the paper can be used with state-space models, in practice it is more convenient to use input-output models (5). Let $u^{sp}(k)$ and $y^{sp}(k)$ correspond to the desired set-point. Because the rudimentary dual-model MPC algorithm described in Section 2 uses the state-space model (14), it is necessary to define the state vector of length $n_x = n_y n_A + n_u \max(n_B - 1, 1)$, whose elements are past input and output values of process variables (Maciejowski, 2002)

$$x(k) = \begin{bmatrix} x_u(k) \\ x_y(k) \end{bmatrix} \tag{25}$$

where

$$x_u(k) = \begin{bmatrix} u(k-1) \\ \vdots \\ u(k-\max(n_B-1,1)) \end{bmatrix}, \quad x_y(k) = \begin{bmatrix} y(k) \\ \vdots \\ y(k-n_A+1) \end{bmatrix} \tag{26}$$

are vectors of length $n_u \max(n_B-1,1)$ and $n_y n_A$, respectively. For simplicity of presentation $n_A = \max(n_A^m)$ where $m = 1,\ldots,n_y$ and $n_B = \max(n_B^{m,n})$ where $m = 1,\ldots,n_y$, $n = 1,\ldots,n_u$.

The key idea of the stable MPC-NPL algorithm is to use two different performance indices. In order to calculate future control moves, the MPC-NPL quadratic programming problem (10) is solved. It uses the classical performance index (2) which takes into account predicted differences from the set-point and future control increments, as it is done in the most popular MPC algorithms based on input-output models, for example DMC (Cutler and Ramaker, 1979) or GPC (Clarke et al., 1987). Thanks to linearisation and free trajectory calculation, the MPC-NPL optimisation problem is convex, the global solution is always found.

The second performance index, named "the test function", is used to decide whether or not the obtained solution should be applied to the system in order to guarantee stability of MPC. Similarly to (19), it is defined as

$$J^t(k) = \sum_{p=0}^{N-1} \theta(x(k+p|k))\big(\|x(k+p|k) - x^{sp}(k)\|^2_{\boldsymbol{Q}} + \|u(k+p|k) - u^{sp}(k)\|^2_{\boldsymbol{R}} \big) \tag{27}$$

where $u^{sp}(k)$ corresponds to the output set-point $y^{sp}(k)$. Considering the structure of the state vector (26), one has

$$x^{sp}(k) = \begin{bmatrix} \bar{u}^{sp}(k) \\ \bar{y}^{sp}(k) \end{bmatrix}, \quad \bar{u}^{sp}(k) = \begin{bmatrix} u^{sp}(k) \\ \vdots \\ u^{sp}(k) \end{bmatrix}, \quad \bar{y}^{sp}(k) = \begin{bmatrix} y^{sp}(k) \\ \vdots \\ y^{sp}(k) \end{bmatrix} \tag{28}$$

and, $\bar{u}^{sp}(k) \in \Re^{n_u \max(n_B-1,1)}$, $\bar{y}^{sp}(k) \in \Re^{n_y n_A}$, $u(k+p|k) = u(k+N_u-1|k)$ for $p \geq N_u$.

In the rudimentary MPC-NPL algorithm (i.e. without guaranteed stability) decision variables are found using quadratic programming approach, in which the objective function is quadratic, the constraints are linear. The optimisation problem (10) is of quadratic programming type. Hence, bearing in mind the rudimentary dual-mode algorithm presented is Section 2, the main issue to address is how to take into account the nonlinear stabilising inequality constraint (16). In the approach presented in this paper the nonlinear constraint is transformed into a set of linear inequality ones. The idea of the method is depicted in

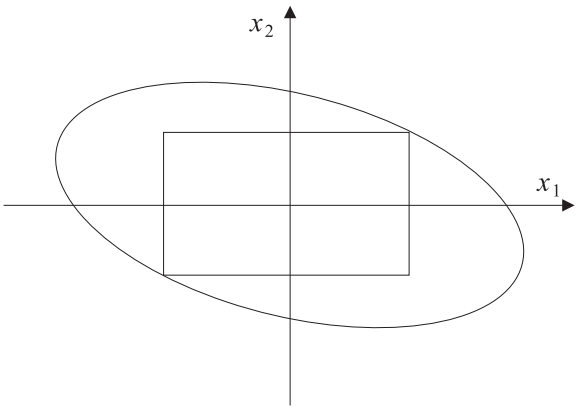

Figure 2. The ellipsoid Ω_α and the resulting box determined by vectors of the standard basis ($n_x = 2$)

Fig. 2 for $n_x = 2$. The ellipsoid is replaced by a set of linear constraints determined by vectors of the standard basis. The resulting box, which is symmetric with respect to the axes, belongs to the interior of the ellipsoid. Hence, if the predicted state $x(k+N|k)$ is in this box, it is in the ellipsoid. The nonlinear constraint (16) is then replaced by linear constraints

$$\tilde{x}^{\min} \leq x(k+N|k) - x^{sp}(k) \leq \tilde{x}^{\max} \tag{29}$$

where, for the sake of symmetry, $\tilde{x}^{\max} = -\tilde{x}^{\min}$.

In order to find the box, i.e. the quantities $\tilde{x}_1^{\max}, \ldots, \tilde{x}_{n_x}^{\max}$, composing the vector $\tilde{x}^{\max}$, it can be noticed that the box has 2^{n_x} vertices, because of the symmetry it is sufficient to take into account only half of them. They can be described as

$$w_i = \left(\sigma_1 \tilde{x}_1^{\max}, \sigma_2 \tilde{x}_2^{\max}, \ldots, \sigma_{n_x-1} \tilde{x}_{n_x-1}^{\max}, \tilde{x}_{n_x}^{\max}\right) \tag{30}$$

where $\sigma_n \in \{-1, 1\}$, $n = 1, \ldots, n_x - 1$, $i = 1, \ldots, 2^{n_x-1}$. The quantities $\tilde{x}_1^{\max}, \ldots, \tilde{x}_{n_x}^{\max}$, which uniquely determine the linear constraints (29), are found from the following optimisation problem

$$\begin{aligned} &\max_{\tilde{x}_1^{\max}, \ldots, \tilde{x}_{n_x}^{\max}} \prod_{n=1}^{n_x} \tilde{x}_n^{\max} \\ &\text{subject to} \\ &\quad \tilde{x}_n^{\max} > 0, \quad n = 1, \ldots, n_x \\ &\quad \|w_i\|_{\boldsymbol{P}}^2 \leq \alpha, \quad i = 1, \ldots, 2^{n_x-1} \end{aligned} \tag{31}$$

because the vertices of the box have to belong to the ellipsoid and the biggest possible ellipsoid has to be found.

If the algorithm uses input-output models, as it is assumed in the paper, it is necessary to reformulate the constraints (29) into corresponding constraints imposed on input and output variables. Using the state vector (26) one has

$$\begin{aligned} \tilde{u}^{\min} &\leq x_u(k+N|k) - \bar{u}^{sp}(k) \leq \tilde{u}^{\max} \\ \tilde{y}^{\min} &\leq x_y(k+N|k) - \bar{y}^{sp}(k) \leq \tilde{y}^{\max} \end{aligned} \tag{32}$$

where

$$\tilde{u}^{\min} = \begin{bmatrix} u^{\min} \\ \vdots \\ u^{\min} \end{bmatrix}, \ \tilde{u}^{\max} = \begin{bmatrix} u^{\max} \\ \vdots \\ u^{\max} \end{bmatrix}, \ \tilde{y}^{\min} = \begin{bmatrix} y^{\min} \\ \vdots \\ y^{\min} \end{bmatrix}, \ \tilde{y}^{\max} = \begin{bmatrix} y^{\max} \\ \vdots \\ y^{\max} \end{bmatrix} \tag{33}$$

$\tilde{u}^{\min}$, $\tilde{u}^{\max} \in \Re^{n_u \cdot \max(n_B-1,1)}$, $\tilde{y}^{\min}$, $\tilde{y}^{\max} \in \Re^{n_y n_A}$. The first group of the above constraints are linear with respect to the future values of the manipulated variables, because $x_u(k+N|k) = \left[u^T(k-n_b+1+N|k) \dots u^T(k-1+N|k)\right]^T$. Hence, they can be directly used in a quadratic programming procedure. What concerns the second ones, analogously to (6), the following is used

$$\tilde{y}^{\min} \leq \tilde{\boldsymbol{G}}(k)\boldsymbol{\Delta u}(k) + \tilde{\boldsymbol{y}}^0(k) - \bar{y}^{sp}(k) \leq \tilde{y}^{\max} \tag{34}$$

where the matrix $\tilde{\boldsymbol{G}}(k)$, of dimensionality $n_y n_A \times n_u N_u$, is

$$\tilde{\boldsymbol{G}}(k) = \begin{bmatrix} \boldsymbol{S}_{N-n_A+1}(k) & \boldsymbol{S}_{N-n_A}(k) & \dots & \boldsymbol{0} \\ \boldsymbol{S}_{N-n_A+2}(k) & \boldsymbol{S}_{N-n_A+1}(k) & \dots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{S}_N(k) & \boldsymbol{S}_{N-1}(k) & \dots & \boldsymbol{S}_{N-N_u+1}(k) \end{bmatrix} \tag{35}$$

and the vector $\tilde{\boldsymbol{y}}^0(k) = \left[(y^0(k+N-n_a+1|k))^T \dots (y^0(k+N|k))^T\right]^T$ is the corresponding free trajectory. Because it depends only on the past, as in the case of the trajectory $\boldsymbol{y}^0(k)$, it can be easily determined from the nonlinear model (5) assuming only influence of the past. Analogously to the calculation of the dynamic matrix $\boldsymbol{G}(k)$ the superposition principle, similar to (6), and a linearised model are used in (34) to calculate the matrix $\tilde{\boldsymbol{G}}(k)$. Even if the nonlinear model is perfect, it may happen that the state vector calculated from the linearisation of the model belongs to the box constraints replacing the ellipsoid, whereas the accurate prediction of the state $x(k+N|k)$ determined from the nonlinear model is outside the terminal set Ω_α. The problem of inaccurate (suboptimal) prediction, which may lead to dissatisfaction of the nonlinear constraint (16) and, in the worst case, to instability, can be solved by decreasing the box replacing the ellipsoid. It is done by using the constraints

$$\tilde{y}^{\min} + \tilde{\varepsilon}^{\min} \leq \tilde{\boldsymbol{G}}(k)\boldsymbol{\Delta u}(k) + \tilde{\boldsymbol{y}}^0(k) - \bar{y}^{sp}(k) \leq \tilde{y}^{\max} - \tilde{\varepsilon}^{\max} \tag{36}$$

where the vectors $\tilde{\varepsilon}^{\min} \geq 0$, $\tilde{\varepsilon}^{\max} \geq 0$ are of length $n_y n_A$.

All things considered, in the stable MPC-NPL algorithm the sequence of future control moves $\Delta \boldsymbol{u}(k)$ is calculated from the following quadratic programming problem

$$\begin{aligned}
&\min_{\Delta \boldsymbol{u}(k),\ \boldsymbol{\varepsilon}^{\min},\ \boldsymbol{\varepsilon}^{\max}} \Big\{ \left\| \boldsymbol{y}^{sp}(k) - \boldsymbol{G}(k)\Delta \boldsymbol{u}(k) - \boldsymbol{y}^{0}(k) \right\|^2_{\boldsymbol{M}} \\
&\qquad + \|\Delta \boldsymbol{u}(k)\|^2_{\boldsymbol{\Lambda}} + \rho^{\min} \left\| \boldsymbol{\varepsilon}^{\min} \right\|^2 + \rho^{\max} \left\| \boldsymbol{\varepsilon}^{\max} \right\|^2 \Big\} \\
&\text{subject to} \\
&\quad \boldsymbol{u}^{\min} \le \boldsymbol{J}\Delta \boldsymbol{u}(k) + \boldsymbol{u}^{k-1}(k) \le \boldsymbol{u}^{\max} \\
&\quad -\Delta \boldsymbol{u}^{\max} \le \Delta \boldsymbol{u}(k) \le \Delta \boldsymbol{u}^{\max} \\
&\quad \boldsymbol{y}^{\min} - \boldsymbol{\varepsilon}^{\min} \le \boldsymbol{y}^{0}(k) + \boldsymbol{G}(k)\Delta \boldsymbol{u}(k) \le \boldsymbol{y}^{\max} + \boldsymbol{\varepsilon}^{\max} \\
&\quad \boldsymbol{\varepsilon}^{\min} \ge 0, \quad \boldsymbol{\varepsilon}^{\max} \ge 0 \\
&\quad \tilde{u}^{\min} \le x_u(k+N|k) - \bar{u}^{sp}(k) \le \tilde{u}^{\max} \\
&\quad \tilde{y}^{\min} + \tilde{\varepsilon}^{\min} \le \tilde{\boldsymbol{G}}(k)\boldsymbol{\Delta u}(k) + \tilde{\boldsymbol{y}}^{0}(k) - \bar{y}^{sp}(k) \le \tilde{y}^{\max} - \tilde{\varepsilon}^{\max}.
\end{aligned} \tag{37}$$

At each sampling instant the above optimisation problem is commenced with $\tilde{\varepsilon}^{\min} = \tilde{\varepsilon}^{\max}=0$. Having found the future control increments, the nonlinear prediction is calculated afterwards using the nonlinear model (5). If $x(k+N|k) \in \Omega_\alpha$ the solution is accepted, otherwise the quantities $\tilde{\varepsilon}^{\min}$ and $\tilde{\varepsilon}^{\max}$ are increased, which results in decreasing the box depicted in Fig. 2 and the optimisation problem (37) is repeated. Although during performing the experiments, whose results are presented in Section 5 it was not necessary to reduce the size of the box replacing the ellipsoid, in general the algorithm should make it possible.

Considering the rudimentary MPC-NPL algorithm and the basic scheme of the suboptimal dual-mode MPC described in Section 2, the proposed stable MPC-NPL algorithm is as follows:

1. Set $\mu \in (0, 1]$.
2. For $k = 0$: if $x(0) \in \Omega_\alpha$ apply $u(0) = \boldsymbol{K}x(0)$. Otherwise, calculate the solution $\Delta \boldsymbol{u}(0)$ to the quadratic optimisation problem (37) and corresponding state trajectory $\boldsymbol{x}(0)$. Apply $u(0) = \Delta u(0|0) + u(k-1)$.
3. For $k \ge 1$: if $x(k) \in \Omega_\alpha$ apply $u(k) = \boldsymbol{K}x(k)$. Otherwise, calculate the solution $\Delta \boldsymbol{u}(k)$ to the quadratic optimisation problem (37) and the corresponding state trajectory $\boldsymbol{x}(k)$. If the condition

$$J^t(k) \le J^t(k-1) - \mu \left(\|x(k-1)\|^2_{\boldsymbol{Q}} + \|u(k-1) - u^{sp}(k)\|^2_{\boldsymbol{R}} \right) \tag{38}$$

is satisfied, apply $u(k) = \Delta u(k|k) + u(k-1)$. If the condition (38) is not satisfied, apply $u(k) = u(k|k-1)$.

At each sampling instant, k, the nonlinear model (5) is used to calculate nonlinear free trajectories $\boldsymbol{y}^0(k)$, $\tilde{\boldsymbol{y}}^0(k)$ and a local linearisation, from which step-response coefficients and resulting dynamic matrices $\boldsymbol{G}(k)$, $\tilde{\boldsymbol{G}}(k)$ are determined.

The linear controller $\boldsymbol{K}$ must not violate the constraints in the optimisation problem (37). In order to determine the size of the set Ω_α (Section 4), they have to be transformed into appropriate sets X, U defining the constraints (21).

4. Calculation of the terminal set Ω_α

As far as the development of the algorithm is concerned, it is necessary to find off-line the terminal set Ω_α and a local stabilising linear controller $\boldsymbol{K}$. Considering the constraints (21) imposed on state and input variables in the MPC optimisation problem (22), the linear controller, although unconstrained, must calculate control values which never violate the constraints, i.e.

$$x(k) \in X \tag{39}$$

$$u(k) = \boldsymbol{K}x(k) \in U. \tag{40}$$

Linearisation of the nonlinear model (14) in the origin is

$$x(k+1) = \boldsymbol{A}x(k) + \boldsymbol{B}u(k) \tag{41}$$

where

$$\boldsymbol{A} = \left.\frac{\partial f(x(k),u(k))}{\partial x(k)}\right|_{x(k)=0,\ u(k)=0}, \quad \boldsymbol{B} = \left.\frac{\partial f(x(k),u(k))}{\partial u(k)}\right|_{x(k)=0,\ u(k)=0}. \tag{42}$$

The matrix $\boldsymbol{K}$ is found, for example, using the LQ approach, so that the closed-loop system

$$x(k+1) = \boldsymbol{A}_c x(k) \tag{43}$$

where $\boldsymbol{A}_c = \boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}$, is asymptotically stable. The terminal set Ω_α is defined by (17), i.e. $\Omega_\alpha = \left\{x(k) \in \Re^{n_x} : \|x(k)\|^2_{\boldsymbol{P}} \leq \alpha\right\}$, where the matrix $\boldsymbol{P} > 0$ is the solution to the Lyapunov equation

$$\boldsymbol{A}_c^T \boldsymbol{P} \boldsymbol{A}_c - \boldsymbol{P} = -(\boldsymbol{Q} + \boldsymbol{K}^T \boldsymbol{R} \boldsymbol{K}). \tag{44}$$

THEOREM 1 *Assuming that the linearisation (41) of the nonlinear system (14) in the origin is stabilisable, there exists a constant $\alpha \in (0, \infty)$ which determines the size of the terminal set Ω_α such that:*

a) $\forall\ x(k) \in \Omega_\alpha \quad x(k) \in X,\ u(k) = \boldsymbol{K}x(k) \in U,$

b) the set Ω_α is invariant for the nonlinear system (14) controlled by the linear control law $u(k) = \boldsymbol{K}x(k)$, i.e.

$$x(k) \in \Omega_\alpha \Longrightarrow x(k+1) = f(x(k), \boldsymbol{K}x(k)) \in \Omega_\alpha. \tag{45}$$

The first property guarantees that the constraints (39) and (40) are satisfied for all $x(k) \in \Omega_\alpha$. The invariance property implies that each trajectory

commencing in the terminal set Ω_α never leaves this set. Ω_α is a positively invariant set for the closed loop system with the local stabilising controller (40). For continuous-time systems, detailed stability considerations equivalent to Theorem 1 and its proof can be found in (Chen and Allgöwer, 1998; Michalska and Mayne, 1993). Here, the discrete-time case is considered.

Proof. Because $(0,0) \in interior(X \times U)$, for a matrix $\boldsymbol{P} > 0$ there exists a constant $\alpha_0 \in (0,\infty)$ such that $\forall\, x(k) \in \Omega_{\alpha_0} \quad x(k) \in X$, $u(k) = \boldsymbol{K}x(k) \in U$. In the set Ω_{α_0} the constraints (39) and (40) are satisfied, but, in general, it may be not invariant.

When the invariant set Ω_α has to be found, it is necessary to consider the trajectory of the nonlinear system (14) controlled by the linear controller $u(k) = \boldsymbol{K}x(k)$, i.e. the system

$$x(k+1) = f(x(k), \boldsymbol{K}x(k)). \tag{46}$$

Using the Lyapunov theory (Kalman and Bertram, 1960) it is necessary to determine how the function

$$V(k) = \|x(k)\|^2_{\boldsymbol{P}} \tag{47}$$

evolves along the trajectory of the system (46), similarly as it is done in (Michalska and Mayne, 1993) for continuous-time systems. Let

$$\phi(k) = f(x(k), \boldsymbol{K}x(k)) - \boldsymbol{A}_c x(k). \tag{48}$$

Hence

$$\lim_{x(k)\to 0} \frac{\|\phi(k)\|^2}{\|x(k)\|^2} = 0. \tag{49}$$

From (48) one has

$$V(k+1) - V(k) = \left\|\phi^T(k)\right\|^2_{\boldsymbol{P}} + 2\phi^T(k)\boldsymbol{P}\boldsymbol{A}_c x(k) + \left\|x^T(k)\right\|^2_{(\boldsymbol{A}_c^T\boldsymbol{P}\boldsymbol{A}_c - \boldsymbol{P})} \tag{50}$$

Using the Lyapunov equation (44) becomes

$$V(k+1) - V(k) = \left\|\phi^T(k)\right\|^2_{\boldsymbol{P}} + 2\phi^T(k)\boldsymbol{P}\boldsymbol{A}_c x(k) - \left\|x^T(k)\right\|^2_{\boldsymbol{Q}_c} \tag{51}$$

where $\boldsymbol{Q}_c = \boldsymbol{A}_c^T\boldsymbol{P}\boldsymbol{A}_c - \boldsymbol{P} = -(\boldsymbol{Q} + \boldsymbol{K}^T\boldsymbol{R}\boldsymbol{K})$. It also holds true that

$$V(k+1) - V(k) \le \left\|\phi^T(k)\right\|^2_{\boldsymbol{P}} + 2\left|\phi^T(k)\boldsymbol{P}\boldsymbol{A}_c x(k)\right| - \left\|x^T(k)\right\|^2_{\boldsymbol{Q}_c}. \tag{52}$$

Considering the right-hand side of inequality (52), it can be noticed that its last part has the dominating absolute value, since it is independent of $\phi(k)$. This is because

- For the first part of the right-hand side of inequality (52), using the inequality

$$\lambda_{\min}(\boldsymbol{X}) \left\|x(k)\right\|^2 \leq \left\|x(k)\right\|^2_{\boldsymbol{X}} \leq \lambda_{\max}(\boldsymbol{X}) \left\|x(k)\right\|^2 \tag{53}$$

where $\lambda_{\min}(\boldsymbol{X})$ and $\lambda_{\max}(\boldsymbol{X})$ are minimum and maximum eigenvalues of a matrix $\boldsymbol{X} > 0$, respectively, one has

$$\lim_{x(k)\to 0} \frac{\left\|\phi(k)\right\|^2_{\boldsymbol{P}}}{\left\|x(k)\right\|^2_{\boldsymbol{Q}_c}} \leq \frac{\lambda_{\max}(\boldsymbol{P})}{\lambda_{\min}(\boldsymbol{Q}_c)} \lim_{x(k)\to 0} \frac{\left\|\phi(k)\right\|^2}{\left\|x(k)\right\|^2} \tag{54}$$

and from (49) it follows that

$$\lim_{x(k)\to 0} \frac{\left\|\phi(k)\right\|^2_{\boldsymbol{P}}}{\left\|x(k)\right\|^2_{\boldsymbol{Q}_c}} = 0. \tag{55}$$

- For the second part of the right-hand side of inequality (52) one has

$$\left|\phi^T(k)\boldsymbol{P}\boldsymbol{A}_c x(k)\right| \leq \left\|\phi(k)\right\| \left\|\boldsymbol{P}\boldsymbol{A}_c x(k)\right\| \leq \left\|\phi(k)\right\| \left\|\boldsymbol{P}\boldsymbol{A}_c\right\| \left\|x(k)\right\|. \tag{56}$$

Because, again using (53)

$$\lim_{x(k)\to 0} \frac{\left|\phi^T(k)\boldsymbol{P}\boldsymbol{A}_c x(k)\right|}{\left\|x(k)\right\|^2_{\boldsymbol{Q}_c}} \leq \frac{\left\|\boldsymbol{P}\boldsymbol{A}_c\right\|}{\lambda_{\min}(\boldsymbol{Q}_c)} \lim_{x(k)\to 0} \frac{\left\|\phi(k)\right\|^2}{\left\|x(k)\right\|^2} \tag{57}$$

and taking into account (49) one has

$$\lim_{x(k)\to 0} \frac{\left|\phi^T(k)\boldsymbol{P}\boldsymbol{A}_c x(k)\right|}{\left\|x(k)\right\|^2_{\boldsymbol{Q}_c}} = 0. \tag{58}$$

Finally, from (55) and (58),

$$\lim_{x(k)\to 0} \frac{\left\|\phi(k)\right\|^2_{\boldsymbol{P}} + 2\left|\phi^T(k)\boldsymbol{P}\boldsymbol{A}_c x(k)\right| - \left\|x(k)\right\|^2_{\boldsymbol{Q}_c}}{\left\|x(k)\right\|^2_{\boldsymbol{Q}_c}} = -1. \tag{59}$$

Let $\kappa \in (0,1)$. If follows that

$$\begin{aligned} &\forall \kappa \in (0,1)\ \exists \alpha > 0\ x(k) \in \Omega_\alpha \backslash \{\boldsymbol{0}\} \Longrightarrow \\ &\qquad -1 \leq \frac{\left\|\phi(k)\right\|^2_{\boldsymbol{P}} + 2\left|\phi^T(k)\boldsymbol{P}\boldsymbol{A}_c x(k)\right| - \left\|x(k)\right\|^2_{\boldsymbol{Q}_c}}{\left\|x(k)\right\|^2_{\boldsymbol{Q}_c}} \leq -\kappa. \end{aligned} \tag{60}$$

The quantity κ is an adjustable parameter, it determines the maximum possible size of the set Ω_α. The bigger κ, the smaller the maximum possible α and the terminal set Ω_α.

Statements (52) and (60) imply that

$$\forall \kappa \in (0,1)\ \exists \alpha > 0 x(k) \in \Omega_\alpha \Longrightarrow V(k+1) - V(k) \leq -\kappa \left\| x(k) \right\|^2_{\boldsymbol{Q}_c} . \quad (61)$$

Because $\boldsymbol{Q} > 0$, if follows that $\boldsymbol{Q}_c > 0$. For a chosen value of the parameter κ, in the resulting terminal set Ω_α the function $V(k)$ can then be the Lyapunov function of the system (43), which is a linearisation of the nonlinear system (14), and of the nonlinear system controlled by the linear control law (46). From the definition of the function $V(k)$, given by (47), and using (61) one can conclude that each trajectory commencing from the set Ω_α, determined by α corresponding to the chosen value of κ, will remain in its interior, the set Ω_α is thus invariant, i.e. $x(k) \in \Omega_\alpha \Longrightarrow x(k+1) \in \Omega_\alpha$. The equilibrium point, the origin, is asymptotically stable for systems (43) and (46). Obviously, the statement (61) guarantees the existence of suitable α also satisfying $\alpha \in (0, \alpha_0)$, so that the set Ω_α has the properties a) and b) from Theorem 1. ■

All things considered, the terminal set Ω_α is found by means of the following procedure:

1. From the linearisation (41) of the nonlinear system (14) calculate the stabilising linear control law $u(k) = \boldsymbol{K}x(k)$.
2. Calculate the matrix $\boldsymbol{P}$, the solution to the Lyapunov equation (44).
3. Calculate the biggest $\alpha_0 \in (0, \infty)$ such that

$$\forall x(k) \in \Omega_{\alpha_0} \quad x(k) \in X, \quad \boldsymbol{K}x(k) \in U. \quad (62)$$

4. Calculate the biggest $\alpha \in (0, \alpha_0)$ such that, for the chosen κ,

$$\forall x(k) \in \Omega_\alpha \left(\left\| \phi(k) \right\|^2_{\boldsymbol{P}} + 2 \left| \phi^T(k) \boldsymbol{P} \boldsymbol{A}_c x(k) \right| + (\kappa - 1) \left\| x(k) \right\|^2_{\boldsymbol{Q}_c} \right) \leq 0. \quad (63)$$

The problem solved in the third step of the above procedure belongs to the class of semi-infinite optimisation, since the constraints have to be satisfied in the whole set Ω_{α_0}. Let $d(x(k), X)$ and $d(\boldsymbol{K}x(k), U)$ denote the distance from $x(k)$ to the set X and the distance from $\boldsymbol{K}x(k)$ to the set U, respectively. The statement (62) is equivalent to

$$\forall x(k) \in \Omega_{\alpha_0} \quad d(x(k), X) = 0, \quad d(\boldsymbol{K}x(k), U) = 0 \ . \quad (64)$$

Of course, constraints (64) are satisfied only when the maximum distance between any point $x(k)$ and the set X is 0 and the maximum distance between any point $\boldsymbol{K}x(k)$ and the set U is 0. Hence, statements (64) can be transformed into the following equivalent statements

$$\max_{\left\| x(k) \right\|^2_{\boldsymbol{P}} \leq \alpha_0} (d(x(k), X)) = 0 \quad \text{and} \quad \max_{\left\| x(k) \right\|^2_{\boldsymbol{P}} \leq \alpha_0} (d(\boldsymbol{K}x(k), U)) = 0 \ , \quad (65)$$

which can be verified by solving two standard optimisation problems with inequality constraints

$$\begin{array}{ll} \min\limits_{x(k)}(-d(x(k),X)) \\ \text{subject to} \\ \quad \|x(k)\|^2_{\boldsymbol{P}} \leq \alpha_0 \end{array} \quad , \quad \begin{array}{ll} \min\limits_{x(k)}(-d(\boldsymbol{K}x(k),U)) \\ \text{subject to} \\ \quad \|x(k)\|^2_{\boldsymbol{P}} \leq \alpha_0 \end{array} \quad . \tag{66}$$

Thus, the problem in step 3 can be rewritten as

$$\begin{array}{l} \max\limits_{\alpha_0} \alpha_0 \\ \text{subject to} \\ \quad \max\limits_{\|x(k)\|^2_{\boldsymbol{P}} \leq \alpha_0} (d(x(k),X)) = 0 \\ \quad \max\limits_{\|x(k)\|^2_{\boldsymbol{P}} \leq \alpha_0} (d(\boldsymbol{K}x(k),U)) = 0 \end{array} \quad . \tag{67}$$

It is obvious that for small values of α_0 the statement (62) is true and also (39), (40) are true. Hence, the parameter α_0 can be found by a recurrent procedure, in which its initial value is increased, the optimisation problems (66) are solved and the signs of the objective functions for the optimal determined solution are checked.

Considering the efficiency of solving the optimisation problems (66), the way the quantities $d(x(k),X)$ and $d(\boldsymbol{K}x(k),U)$ are calculated is crucial. In general, the distance from the point $x(k)$ to the set X is given by

$$d(x(k),X) = \left\|x(k) - x^0(k)\right\|, \tag{68}$$

where $x^0(k)$ is the solution to the following optimisation problem

$$\begin{array}{l} \min\limits_{x^*(k)} \|x(k) - x^*(k)\| \\ \text{subject to} \\ \quad x^*(k) \in X \end{array} \quad . \tag{69}$$

Analogously, the distance to the set U is given by

$$d(\boldsymbol{K}x(k),U) = \left\|\boldsymbol{K}x(k) - u^0(k)\right\|. \tag{70}$$

where $u^0(k)$ is the solution to the following optimisation problem

$$\begin{array}{l} \min\limits_{u^*(k)} \|\boldsymbol{K}x(k) - u^*(k)\| \\ \text{subject to} \\ \quad u^*(k) \in U \end{array} \quad . \tag{71}$$

The formulae presented so far are very general. In the most common case of box constraints

$$X = \left\{x(k) \in \Re^{n_x} : x^{\min} \leq x(k) \leq x^{\max}\right\} \tag{72}$$

$$U = \left\{u(k) \in \Re^{n_u} : u^{\min} \leq u(k) \leq u^{\max}\right\} \tag{73}$$

where

$$x^{\min} \leq 0 \leq x^{\max}, \quad u^{\min} \leq 0 \leq u^{\max} \tag{74}$$

the value of α_0, which determines the size of the set Ω_{α_0}, can be found analytically, without any optimisation.

THEOREM 2 *If the state constraints are given in the box form (72),* $\max\limits_{\Omega_\alpha \subset X} \alpha = \max \alpha_0^n$, *where*

$$\alpha_0^n = \frac{\min\left((x_n^{\min})^2, (x_n^{\max})^2\right)}{e_n^T \boldsymbol{P}^{-1} e_n} \tag{75}$$

and e_n is a vector of length n_x, its n^{th} coordinate is 1, the remaining ones are 0.

THEOREM 3 *If the input constraints are given in the box form (73) and the matrix $\boldsymbol{K}$ has full rank,* $\max\limits_{\boldsymbol{K}\Omega_\alpha \subset U} \alpha = \max \alpha_0^n$, *where*

$$\alpha_0^n = \frac{\min\left((u_n^{\min})^2, (u_n^{\max})^2\right)}{e_n^T \boldsymbol{K}\boldsymbol{P}^{-1}\boldsymbol{K}^T e_n} \tag{76}$$

and e_n is a vector of length n_u, its n^{th} coordinate is 1, the remaining ones are 0.

Proofs of Theorems 2 and 3 are given in Appendices A and B, respectively.

From Theorem 2 and Theorem 3 it follows that α_0 calculated in step 3 of the procedure in which the set Ω_α is found, in the case of box constraints imposed on state (72) and input (73) variables, is given by

$$\alpha_0 \in \left(\min_{n=1,\ldots,n_x} \frac{\min\left((x_n^{\min})^2, (x_n^{\max})^2\right)}{e_n^T \boldsymbol{P}^{-1} e_n}, \min_{n=1,\ldots,n_u} \frac{\min\left((u_n^{\min})^2, (u_n^{\max})^2\right)}{e_n^T \boldsymbol{K}\boldsymbol{P}^{-1}\boldsymbol{K}^T e_n} \right). \tag{77}$$

To sum up, if the state and input constraints are given in their general form (21), the value of the parameter α_0, which determines the size of the set Ω_{α_0}, is found by solving the semi-infinite optimisation problem (62), formulated also as the equivalent statement (65). Values of the respective constraint functions are calculated as solutions to two standard minimisation problems with nonlinear inequality constraints (66), the resulting standard optimisation problem is then (67). The distances, which are used as objective functions in (66), are calculated from (68) and (70) using the solutions to the additional optimisation problems (69) and (71). Hence, the resulting optimisation problems are nested. If the constraints are given in box form (72) and (73), the value of α_0 can be calculated analytically from Theorems 2 and 3 without the necessity of solving any optimisation problems.

To guarantee the invariance property of the terminal set Ω_α (Theorem 1), the biggest allowed $\alpha \in (0, \alpha_0)$ is found so that (63) is true for the chosen value of κ. The resulting semi-infinite optimisation problem can be expressed as

$$\begin{aligned} &\max_{\alpha} \alpha \\ &\text{subject to} \\ &\left(\max_{\|x(k)\|^2_{\boldsymbol{P}} \leq \alpha} \left((\kappa - 1) \|x(k)\|^2_{\boldsymbol{Q}_c} + \|\phi(k)\|^2_{\boldsymbol{P}} + 2 \left| \phi^T(k) \boldsymbol{P} \boldsymbol{A}_c x(k) \right| \right) \right) \leq 0 \end{aligned} \tag{78}$$

where the value of the constraint function is calculated as the solution to the following standard optimisation problem with inequality constraint

$$\begin{aligned} &\min_{x(k)} \left((1 - \kappa) \|x(k)\|^2_{\boldsymbol{Q}_c} - \|\phi(k)\|^2_{\boldsymbol{P}} - 2 \left| \phi^T(k) \boldsymbol{P} \boldsymbol{A}_c x(k) \right| \right) \\ &\text{subject to} \\ &\quad \|x(k)\|^2_{\boldsymbol{P}} \leq \alpha \end{aligned} \quad . \tag{79}$$

Analogously to the iterative procedure in which α_0 is found by increasing its value and the optimisation problems (66) are solved, when α has to be found its initial value is increased, the problem (79) is solved, and the sign of the objective function is checked for the optimal solution determined. Naturally, one has take into account that α is restricted to $(0, \alpha_0)$, otherwise the linear controller $u(k) = \boldsymbol{K} x(k)$ may violate the constraints (39) and (40).

5. Experiments

The plant under consideration is a high-purity, high-pressure (1.93 MPa) ethylene-ethane distillation column the structure of which is shown in Fig. 3 (Ławryńczuk and Tatjewski, 2006; Tatjewski, 2007; Tatjewski and Ławryńczuk, 2006). The feed stream consists of ethylene (approx. 80 %), ethane and traces of hydrogen, methane and propylene. The product of the distillation is ethylene which, according to technological requirements, can contain up to 1000 ppm (parts per million) of ethane. The objective is to develop a supervisory controller which would be able to increase relatively fast the impurity level when the composition changes in the feed stream are relatively small. Reducing the purity of the product, of course taking into account the technological limit, results in decreasing energy consumption. Production scale is very big, nominal value of the product stream is 43 tons/h. The column has 121 trays, the feed stream is delivered to the tray number 37.

Two fast single-loop PID controllers (denoted as LC) are used to stabilise the levels in reflux tank and bottom product tank. Yet another PID controller (denoted as TC) is used to control the temperature on the tray number 13. The PID controllers comprise the basic control layer. As far as the supervisory MPC algorithm is concerned, the control loop has one manipulated variable r, which is

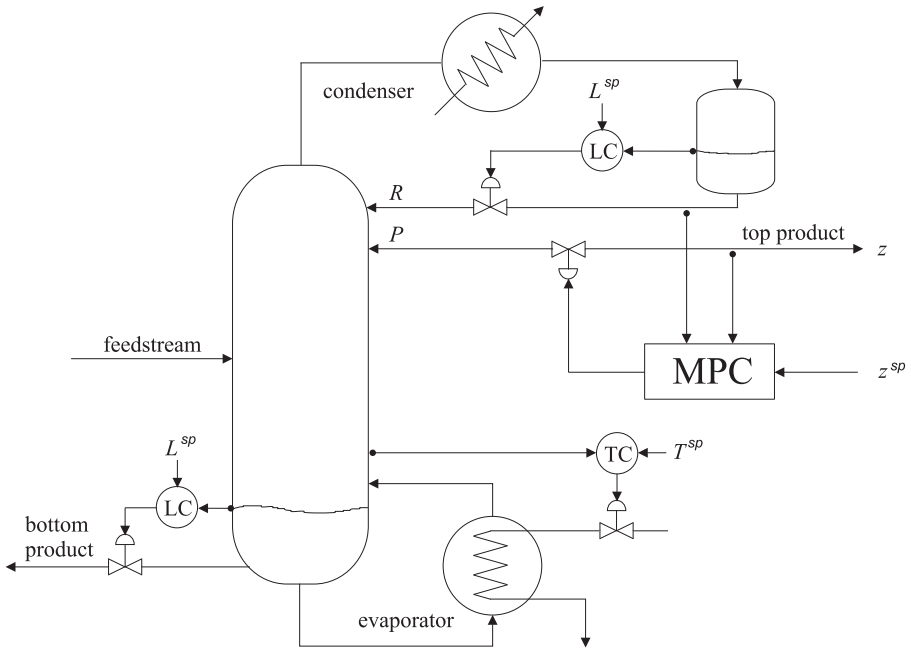

Figure 3. High-purity ethylene-ethane distillation column control system structure

the reflux ratio $r = \frac{R}{P}$, where R and P are reflux and product stream flow rates, respectively, and one controlled variable z, which represents the impurity of the product. In fact, the desired reflux ratio, calculated by the supervisory MPC algorithm, is enforced by manipulating the product flow rate. The reflux stream is delivered to the column by the top tray and the product stream is taken from the tray number 110. Sample time of the supervisory MPC algorithms is 40 minutes (slow composition analyser.)

Four models of the plant are used. The fundamental model is used as the real process during simulations. This model is simulated open-loop in order to obtain two sets of data, namely training and test data sets. Both sets contain 2000 samples, the output signal contains noise. An identification procedure is carried out, as a result three empirical models are obtained:

a) a linear model for low impurity level (100 ppm)

$$y(k) = b_3^{low} u(k-3) - a_1^{low} y(k-1), \tag{80}$$

a) a linear model for high impurity level (850 ppm)

$$y(k) = b_3^{high} u(k-3) - a_1^{high} y(k-1), \tag{81}$$

c) a neural model

$$y(k) = f^{NARX}(u(k-3), y(k-1))), \tag{82}$$

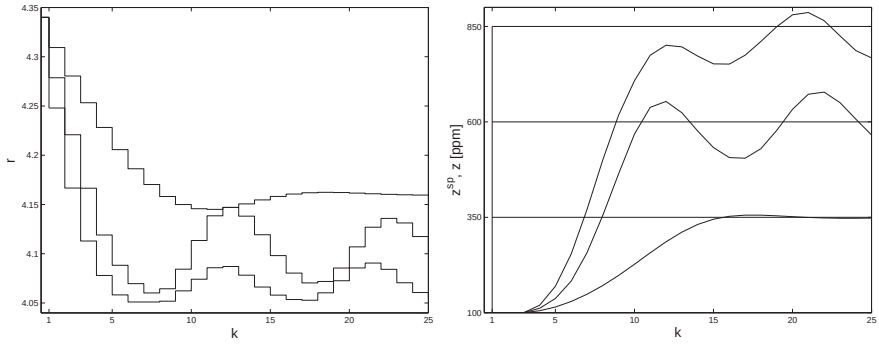

Figure 4. Simulation results of the MPC algorithm based on the linear model valid for low impurity level

All three empirical models have the same arguments, determined by $n_A = 1$, $\tau = n_B = 3$. Because input and output process variables are of different order, they are scaled as $u = \frac{10}{3}(r - r_0)$, $y = 0.001(z - z_0)$ where $r_0 = 4.34$, $z_0 = 100$ correspond to the initial operating point. A Multi Layer Perceptron (MLP) neural model used. It contains one hidden layer with hyperbolic tangent transfer functions and a linear output, 6 neurons in the hidden layer are used.

The compared MPC strategies are:

a) the linear MPC algorithm based on the linear model (80) for low impurity level,

b) the linear MPC algorithm based on the linear model (81) for high impurity level,

c) the rudimentary (i.e. without guaranteed stability) suboptimal nonlinear MPC-NPL algorithm based on the neural model (82),

d) the suboptimal stable nonlinear MPC-NPL algorithm based on the same neural model (82).

The horizons are $N = 10$, $N_u = 3$, the weighting matrices $\boldsymbol{M}_p = 1$ and $\boldsymbol{\Lambda}_p = 2$. Three set-point changes are used, at the sampling instant $k = 1$ the set-point value is changed from 100 ppm to 350 ppm, 600 ppm and 850 ppm, respectively. Because of technological reasons the following constraints are imposed on the reflux ratio: $r^{min} = 4.051$, $r^{max} = 4.4571$.

At first, MPC algorithms based on two linear models are developed. The first one is valid for low impurity level and the resulting control algorithm works well in this region but exhibits unacceptable oscillatory behaviour for medium and big set-point changes as it is shown in Fig. 4. The second linear model captures the process properties for high impurity level and the closed-loop response is fast enough for the biggest set-point change, but is very slow for smaller ones,

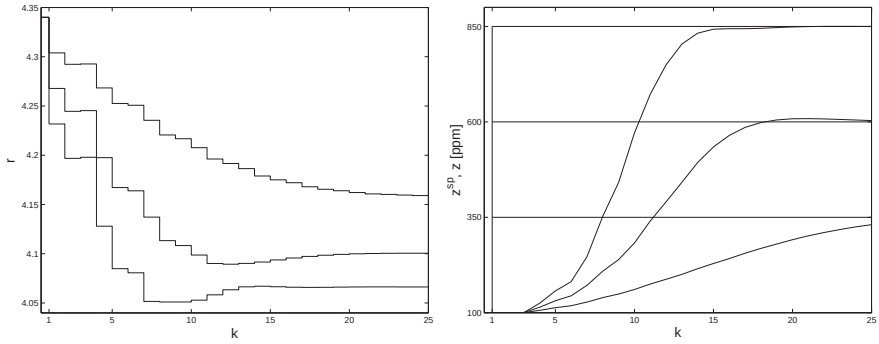

Figure 5. Simulation results of the MPC algorithm based on the linear model valid for high impurity level

as it is shown in Fig. 5. Simulation results of the MPC-NPL algorithm with the neural network model are depicted in Fig. 6. The suboptimal algorithm is stable and fast for all three set-point changes. As demonstrated in (Ławryńczuk and Tatjewski, 2006; Tatjewski, 2007; Tatjewski and Ławryńczuk, 2006), the closed-loop performance obtained in the suboptimal MPC-NPL algorithm is close to that obtained in the computationally prohibitive MPC algorithm with Nonlinear Optimisation (MPC-NO).

The suboptimal nonlinear MPC-NPL algorithm works in practice, but its closed-loop stability is not guaranteed. In order to compare the MPC algorithms without and with guaranteed stability, described in this paper, the weighting matrix $\mathbf{\Lambda}_p = 0.01$ in the objective function (37) is used. Simulation results are depicted in Fig. 7. At the sampling instant $k = 1$ the set-point value is changed from 100 ppm to 600 ppm. The first algorithm becomes unstable, whereas the second one works properly. Of course, such a small value of $\mathbf{\Lambda}_p$ is practically unacceptable, since it results in fast output profile which would be too dangerous from the technological point of view.

Fig. 8 presents simulation results of the MPC algorithm with guaranteed stability with nominal length of the control horizon $N_u = 3$. The weighting matrix $\mathbf{\Lambda}_p$ is set to its nominal value 2. It can be noticed that the bigger the parameter μ the bigger the possibility of violating the condition (38), previously calculated controls are applied to the plant more frequently. When $z^{sp} = 350$ ppm and $z^{sp} = 850$ ppm and when $\mu = 0.01$, the condition (38) is always satisfied, whereas when $\mu = 1$ the solution to the optimisation problem (37) is rejected at $k = 4, 5$. When $z^{sp} = 600$ ppm and when $\mu = 0.01$ the solution is rejected at $k = 3$, when $\mu = 1$ at $k = 3, 4$.

Fig. 9 presents simulation results of the MPC algorithm with guaranteed stability with long control horizon $N_u = 10$. When compared to the shorter horizon case, the influence of the parameter μ is even smaller. For all three

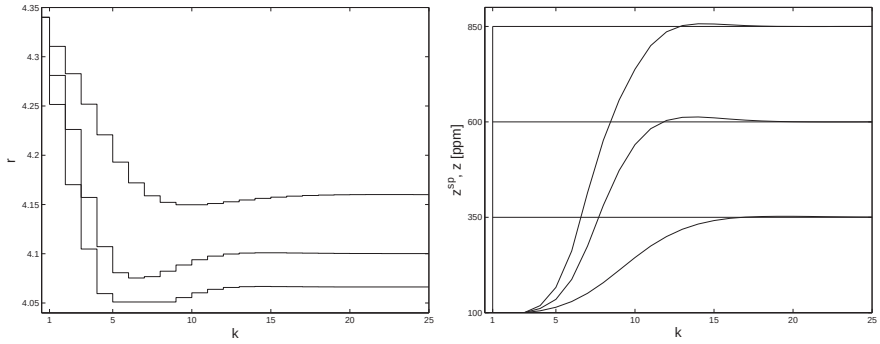

Figure 6. Simulation results of the rudimentary (i.e. without guaranteed stability) MPC-NPL algorithm based on the neural network model

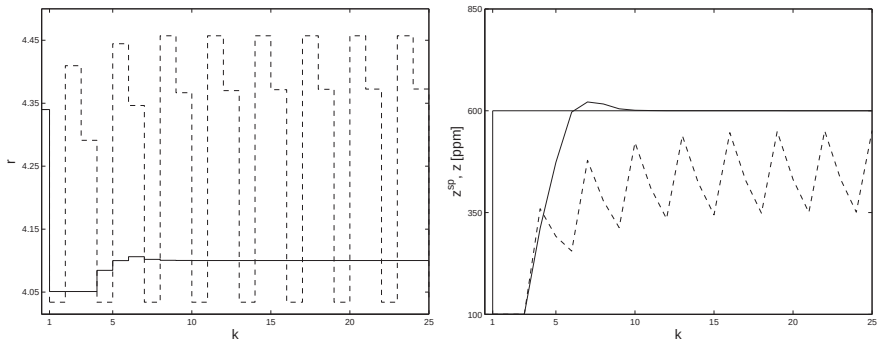

Figure 7. Simulation results of two MPC-NPL algorithms, based on the neural model: without (*dashed line*) and with guaranteed stability (*solid line*), $\mathbf{\Lambda}_p = 0.01$, $\mu = 0.01$

set-point changes and $\mu = 0.01$ the algorithm never uses the previously found sequence. When $\mu = 1$ and $z^{sp} = 350$ ppm the solution is rejected at $k = 6, 7$, when $z^{sp} = 600$ ppm at $k = 4$, when $z^{sp} = 850$ ppm at $k = 5$.

6. Conclusions

The algorithm presented in this paper makes an attempt to bridge the gap between practice and theory. On the one hand, different versions of suboptimal MPC algorithms with on-line linearisation and quadratic programming are widely used in practice. Their stability is practically achieved by tuning horizon lengths and penalty factors. On the other hand, algorithms with guaranteed stability need on-line nonlinear optimisation. The described algorithm results in an easy to solve on-line quadratic programming problem and its closed-loop stability is guaranteed.

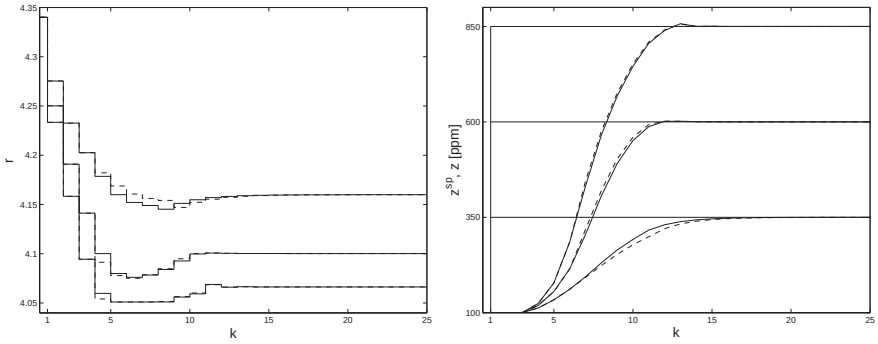

Figure 8. Simulation results of the MPC-NPL algorithm with guaranteed stability, based on the neural model: $\mu = 0.01$ (*dashed line*), $\mu = 1$ (*solid line*), $N_u = 3$

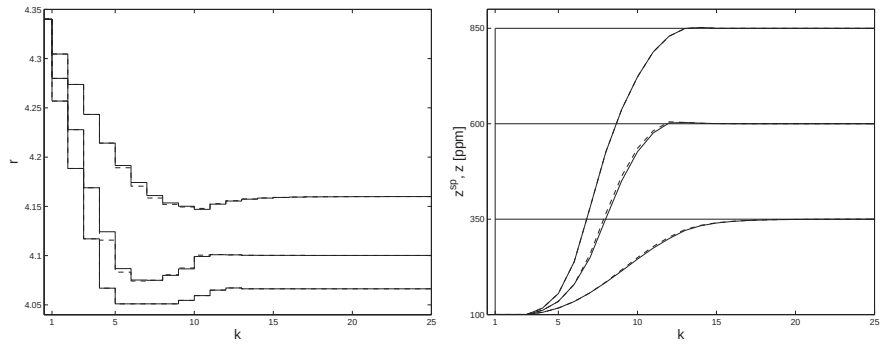

Figure 9. Simulation results of the MPC-NPL algorithm with guaranteed stability, based on the neural model: $\mu = 0.01$ (*dashed line*), $\mu = 1$ (*solid line*), $N_u = 10$

The described suboptimal MPC algorithm has the following advantages: computational efficiency and guaranteed stability. The algorithm combines two ideas: the MPC with Nonlinear Prediction and Linearisation (MPC-NPL) scheme which uses on-line quadratic programming (Ławryńczuk, 2007; Tatjewski, 2007; Tatjewski and Ławryńczuk, 2006) and a modified dual-mode approach (Michalska and Mayne, 1993; Scokaert et al., 1999), for which feasibility of the nonlinear optimisation problem, rather than optimality, implies stability. Contrary to the classical dual-mode MPC, in which nonlinear optimisation is used, the proposed algorithm uses on-line linearisation and computationally reliable quadratic programming to calculate future controls. To keep all the constraints linear, the stabilising nonlinear inequality constraint enforcing bringing the pre-

dicted state at the end of the prediction horizon to the terminal set is transformed into a set of linear ones.

The described algorithm uses input-output NARX models, for example neural networks of various kinds. The algorithm does not restrict the structure of the model. The paper specifies the calculation of the terminal set for different cases of constraints, particularly for box constraints. These methods can be also used in any dual-mode algorithm with state-space models.

Appendices

A. Proof of Theorem 2

To prove Theorem 2 it is sufficient to find the maximum value of α_0 for which the ellipsoid Ω_{α_0} is in the state box constraints (72). Because of the symmetry of the ellipsoid the value α_0 to be found is the smallest of the quantities α_0^n, which satisfy the condition

$$\max_{x^T \boldsymbol{P} x = \alpha_0^n} x_n = \min\left(\left|x_n^{\min}\right|, \left|x_n^{\max}\right|\right) \tag{83}$$

where, for the sake of simplicity, $x = x(k)$. The solution to the above problem, the value $x^* = argmax(x_n)$, is uniquely determined by the optimality conditions

$$x^T \boldsymbol{P} x = \alpha_0^n, \quad \lambda \nabla f(x) = 2\boldsymbol{P} x \tag{84}$$

where λ is an appropriate Lagrange multiplier. Since $f(x) = x_n$

$$\nabla f(x) = e_n. \tag{85}$$

Because the maximum value of x_n has to be found, a pair of (x, λ) is chosen satisfying the above conditions with $\lambda > 0$.

Rearranging the optimality conditions (84) one has

$$\begin{cases} x^T \boldsymbol{P} x = \alpha_0^n \\ 2\boldsymbol{P} x = \lambda e_n \end{cases} \quad \Longrightarrow \quad x = \frac{\lambda}{2} \boldsymbol{P}^{-1} e_n \tag{86}$$

and taking into account that the matrix $\boldsymbol{P} > 0$ is symmetric, it follows that

$$\alpha_0^n = x^T \boldsymbol{P} x = \frac{1}{2} x^T 2\boldsymbol{P} x = \frac{\lambda}{2} x^T e_n \tag{87}$$

$$= \frac{\lambda^2}{4} e_n^T \left(\boldsymbol{P}^{-1}\right)^T e_n = \frac{\lambda^2}{4} e_n^T \boldsymbol{P}^{-1} e_n \tag{88}$$

hence

$$\lambda = \sqrt{\frac{4\alpha_0^n}{e_n^T \boldsymbol{P}^{-1} e_n}}. \tag{89}$$

Using (86) the optimal point is found

$$x^* = \sqrt{\frac{\alpha_0^n}{e_n^T \boldsymbol{P}^{-1} e_n}} \boldsymbol{P}^{-1} e_n. \tag{90}$$

To satisfy (83), i.e.

$$x_n = e_n^T \sqrt{\frac{\alpha_0^n}{e_n^T \boldsymbol{P}^{-1} e_n}} \boldsymbol{P}^{-1} e_n = \sqrt{\alpha_0^n} \sqrt{e_n^T \boldsymbol{P}^{-1} e_n} \tag{91}$$

$$= \min\left(\left|x_n^{\min}\right|, |x_n^{\max}|\right) \tag{92}$$

the value of the parameter α_0^n has to satisfy

$$\alpha_0^n = \frac{\min\left((x_n^{\min})^2, (x_n^{\max})^2\right)}{e_n^T \boldsymbol{P}^{-1} e_n} \tag{93}$$

where the denominator is the n^{th} diagonal element of the matrix $\boldsymbol{P}^{-1}$. Having determined n_x quantities α_0^n, $n = 1, \ldots, n_x$, the maximum value of the parameter α_0 is equal to $\min\limits_{n=1,\ldots,n_x} \alpha_0^n$.

B. Proof of Theorem 3

To prove Theorem 3 it is sufficient to find the maximum value of α_0, for which the ellipsoid of possible controls

$$\varepsilon = \{\boldsymbol{K} x(k) \in \Re^{n_u} : x(k) \in \Omega_{\alpha_0}\} \tag{94}$$

is in the input box constraints (73). The ellipsoid ε is the image of the ellipsoid Ω_{α_0} under the map $\boldsymbol{K}$. It can be easily verified that the surface of the ellipsoid Ω_{α_0}, which is described by the equation $x^T \boldsymbol{P} x = \alpha_0$, is the image of the sphere

$$\theta = \left\{z \in \Re^{n_x} : z^T z = \alpha_0\right\} \tag{95}$$

under the nonsingular map

$$x \to \boldsymbol{S} x \tag{96}$$

with $\boldsymbol{S}$ satisfying

$$\boldsymbol{S}^{-T} \boldsymbol{S}^{-1} = \boldsymbol{P} \quad \left(\Longleftrightarrow \boldsymbol{P}^{-1} = \boldsymbol{S}\boldsymbol{S}^T\right) \tag{97}$$

because, if $\boldsymbol{S}^{-1} x = z$ belongs to θ, $\alpha_0 = z^T z = x^T (\boldsymbol{S}^{-1})^T \boldsymbol{S}^{-1} x$.

Because the matrix $\boldsymbol{P}$ is symmetric, positive definite, it can be expressed as

$$\boldsymbol{P} = \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^T \tag{98}$$

where the matrix $\boldsymbol{U}$ is orthogonal whereas the diagonal matrix $\boldsymbol{\Lambda}$ comprises positive eigenvalues of the matrix $\boldsymbol{P}$. One has

$$\boldsymbol{P}^{-1} = \boldsymbol{U}\boldsymbol{\Lambda}^{-1}\boldsymbol{U}^T = \boldsymbol{U}\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{U}^T \tag{99}$$

$$= \boldsymbol{U}\boldsymbol{\Lambda}^{-\frac{1}{2}}\left(\boldsymbol{U}\boldsymbol{\Lambda}^{-\frac{1}{2}}\right)^T. \tag{100}$$

Hence, let

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Lambda}^{-\frac{1}{2}}. \tag{101}$$

The ellipsoid ε is the image of the sphere θ under the map

$$x \to \boldsymbol{K}\boldsymbol{S}x = \boldsymbol{K}\boldsymbol{U}\boldsymbol{\Lambda}^{-\frac{1}{2}}x. \tag{102}$$

Assuming that the matrix $\boldsymbol{K}$ is of full rank, i.e. $rank(\boldsymbol{K}) = n_u$, the matrix $\boldsymbol{K}\boldsymbol{S}$ of the map (102) can be expressed using Singular Value Decomposition (SVD)

$$\boldsymbol{K}\boldsymbol{S} = \tilde{\boldsymbol{U}}\boldsymbol{\Sigma}\boldsymbol{V}^T = \tilde{\boldsymbol{U}}\left[\begin{array}{cc}\tilde{\boldsymbol{\Sigma}} & \boldsymbol{0}\end{array}\right]\boldsymbol{V}^T \tag{103}$$

where the matrices $\tilde{\boldsymbol{U}}$ and $\boldsymbol{V}$, of dimensions $n_u \times n_u$ and $n_x \times n_x$, respectively, are orthogonal, whereas the nonsingular diagonal matrix $\tilde{\boldsymbol{\Sigma}}$ comprises singular values of the matrix $\boldsymbol{K}\boldsymbol{S}$.

The point y belonging to the ellipsoid ε and its any preimage z belonging to the sphere θ are related by the equation

$$y = \boldsymbol{K}\boldsymbol{U}\boldsymbol{\Lambda}^{-\frac{1}{2}}z = \tilde{\boldsymbol{U}}\left[\begin{array}{cc}\tilde{\boldsymbol{\Sigma}} & \boldsymbol{0}\end{array}\right]\boldsymbol{V}^T z \tag{104}$$

$$= \tilde{\boldsymbol{U}}\left[\begin{array}{cc}\tilde{\boldsymbol{\Sigma}} & \boldsymbol{0}\end{array}\right]\left[\begin{array}{c}\tilde{z}_1\\ \tilde{z}_2\end{array}\right] \tag{105}$$

from which it follows that the component $\tilde{z}_1$ of the vector $\boldsymbol{V}^T z$ satisfies

$$\tilde{z}_1 = \tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{U}}^{-1}y = \tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{U}}^T y. \tag{106}$$

It is also true that

$$\alpha_0 = z^T z = \left(\boldsymbol{V}^T z\right)^T\left(\boldsymbol{V}^T z\right) = \left[\begin{array}{cc}\tilde{z}_1^T & \tilde{z}_2^T\end{array}\right]\left[\begin{array}{c}\tilde{z}_1\\ \tilde{z}_2\end{array}\right] \tag{107}$$

$$= \left[\begin{array}{cc} y^T\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}^{-1} & \tilde{z}_2^T\end{array}\right]\left[\begin{array}{c}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{U}}^T y\\ \tilde{z}_2\end{array}\right] \tag{108}$$

$$= y^T\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\boldsymbol{U}}^T y + \tilde{z}_2^T\tilde{z}_2. \tag{109}$$

Hence, it is easy to see that the point y belongs to the ellipsoid ε if and only if

$$y^T\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}^{-2}\tilde{\boldsymbol{U}}^T y \leq \alpha_0. \tag{110}$$

The value of the parameter α_0, for which the whole of the ellipsoid ε is in the box constraints (73), can be calculated enforcing that only the surface of the ellipsoid satisfies the same constraints. The maximum value of α_0 is determined analogously as in the state box constraints case (Theorem 2). This value is equal to $\min\limits_{n=1,\ldots,n_u} \alpha_0^n$,

$$\alpha_0^n = \frac{\min\left((u_n^{\min})^2, (u_n^{\max})^2\right)}{e_n^T \left(\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}^{-2}\tilde{\boldsymbol{U}}^T\right)^{-1} e_n}. \tag{111}$$

Using the formulae (98) and (101), it can be noticed that

$$\boldsymbol{K}\boldsymbol{P}^{-1}\boldsymbol{K}^T = \boldsymbol{K}\boldsymbol{U}\boldsymbol{\Lambda}^{-1}\boldsymbol{U}^T\boldsymbol{K}^T = \boldsymbol{K}\boldsymbol{U}\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{U}^T\boldsymbol{K}^T \tag{112}$$

$$= (\boldsymbol{K}\boldsymbol{S})(\boldsymbol{K}\boldsymbol{S})^T. \tag{113}$$

Hence, taking into account (103), one has

$$\boldsymbol{K}\boldsymbol{P}^{-1}\boldsymbol{K}^T = \tilde{\boldsymbol{U}} \begin{bmatrix} \tilde{\boldsymbol{\Sigma}} & \boldsymbol{0} \end{bmatrix} \boldsymbol{V}^T\boldsymbol{V} \begin{bmatrix} \tilde{\boldsymbol{\Sigma}} \\ \boldsymbol{0} \end{bmatrix} \tilde{\boldsymbol{U}}^T = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}^2\tilde{\boldsymbol{U}}^T. \tag{114}$$

Taking advantage of the orthogonality of the matrix $\tilde{\boldsymbol{U}}$, one obtains a subexpression of (102)

$$\left(\boldsymbol{K}\boldsymbol{P}^{-1}\boldsymbol{K}^T\right)^{-1} = \left(\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}^2\tilde{\boldsymbol{U}}^T\right)^{-1} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}^{-2}\tilde{\boldsymbol{U}}^T \tag{115}$$

so the values of the parameter α_0^n can be rewritten as

$$\alpha_0^n = \frac{\min\left((u_n^{\min})^2, (u_n^{\max})^2\right)}{e_n^T\boldsymbol{K}\boldsymbol{P}^{-1}\boldsymbol{K}^T e_n} \tag{116}$$

where the denominator is the n^{th} diagonal element of the matrix $\boldsymbol{K}\boldsymbol{P}^{-1}\boldsymbol{K}^T$. Having determined n_u quantities α_0^n, $n = 1, \ldots, n_u$, we obtain the maximum value of the parameter α_0 as equal to $\min\limits_{n=1,\ldots,n_u} \alpha_0^n$.

References

Bemporad, A. (2002) A predictive controller with artificial Lyapunov function for linear systems with input/state constraints. *Automatica* **38**, 1255–1260.

Bemporad, A., Chisci, L. and Mosca, E. (1994) On the stabilizing property of SIORHC. *Automatica* **30**, 2013–2015.

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

Chisci, L., Lombardi, A. and Mosca, E. (1996) Dual-receding horizon control of constrained discrete time systems. *European Journal of Control* **2**, 278–285.

Chisci, L. and Mosca, E. (1994) Stabilizing I-O receding horizon control of CARMA plants. *IEEE Transactions on Automatic Control* **39**, 614–618.

Clarke, D.W. and Scattolini, R. (1991) Constrained receding-horizon predictive control. *Proceedings IEE, Part D* **138**, 347–354.

Clarke, D.W. and Mohtadi, C. (1989) Properties of generalized predictive control. *Automatica* **25**, 859–875.

Clarke, D.W., Mohtadi, C. and Tuffs, P.S. (1987) Generalized predictive control – I. The basic algorithm, II. Extensions and interpretations. *Automatica* **23**, 137–160.

Cutler, R. and Ramaker, B.L. (1979) Dynamic matrix control – a computer control algorithm. *Proceedings of AIChE National Meeting*, Houston, USA.

Declercq, F. and de Keyser, R. (1999) Suboptimal nonlinear predictive controllers. *International Journal of Applied Mathematics and Computer Science* **9**, 129–148.

Garcia, C.E. (1984) Quadratic dynamic matrix control of nonlinear processes: an application to a batch reactor process. *Proceedings of AIChE National Meeting*, San Francisco, USA.

Gattu, G. and Zafiriou, E. (1992) Nonlinear Quadratic Dynamic Matrix Control with state estimation. *Industrial and Engineering Chemistry Research* **31**, 1096–1104.

Henson, M.A. (1998) Nonlinear model predictive control: current status and future directions. *Computers and Chemical Engineering* **23**, 187–202.

Kalman, R.E. and Bertram, J.E. (1960) Control system analysis and design via the "second method" of Lyapunov: I. Continuous-time systems, II. Discrete-time systems. *Journal of Basic Engineering Transactions ASME* **82**, 371–400.

Keerthi, S.S. and Gilbert, E.G. (1988) Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: stability and moving-horizon approximations. *Journal of Optimization Theory and Applications* **57**, 265–293.

Lee, J.H. and Ricker, N.L. (1994) Extended Kalman filter based nonlinear model predictive control. *Industrial and Engineering Chemistry Research* **33**, 1530–1541.

Li, W.C. and Biegler, T. (1989) Multistep, Newton-type control strategies for constrained, nonlinear systems. *Chemical Engineering Research and Design* **67**, 562–577.

Ławryńczuk, M. (2007) A family of model predictive control algorithms with artificial neural networks. *International Journal of Applied Mathematics and Computer Science* **17**, 217–232.

Ławryńczuk, M. and Tatjewski, P. (2006) An efficient nonlinear predic-

tive control algorithm with neural models and its application to a high-purity distillation process. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M., eds., *Lecture Notes in Artificial Intelligence,* **4029**: *The* 8^{th} *International Conference on Artificial Intelligence and Soft Computing, ICAISC 2006, Zakopane, Poland*, Springer, Heidelberg, 76-85.

Ławryńczuk, M. and Tatjewski, P. (2004) A stable dual-mode type nonlinear predictive control algorithm based on on-line linearisation and quadratic programming. *Proceedings of the* 10^{th} *International Conference on Methods and Models in Automation and Robotics*, Midzyzdroje, Poland, 503–510.

Maciejowski, J.A. (2003) *Predictive Control with Constraints.* Prentice Hall, Harlow.

Magni, L., De Nicolao, G., Magnani, L. and Scattolini, R. (2001) A stabilizing model-based predictive control algorithm for nonlinear systems. *Automatica* **37**, 1351–1362.

Marusak, P. and Tatjewski, P. (2003) Stable, effective fuzzy DMC algorithms with on-line quadratic optimisation. *Proceeding of the American Control Conference*, Denver, USA, 3513–3518.

Mayne, D.Q., Rawlings, J.B., Rao, C.V. and Scokaert, P.O.M. (2000) Constrained model predictive control: stability and optimality. *Automatica* **36**, 789–814.

Mayne, D.Q. and Michalska, H. (1990) Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control* **35**, 814–824.

Meadows, E.S., Henson, M.A., Eaton, J.W. and Rawlings, J.B. (1995) Receding horizon control and discontinuous state feedback stabilization. *International Journal of Control* **62**, 1217–1229.

Meadows, E.S., and Rawlings, J.B. (1993) Receding horizon control with an infinite horizon. *Proceedings of the American Control Conference*, San Francisco, USA, 2926–2930.

Michalska, H. and Mayne, D.Q. (1993) Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control* **38**, 1623–1633.

Mollov, S., Babuška, R., Abonyi, J. and Verbruggen, H.B. (2004) Effective Optimization for Fuzzy Model Predictive Control. *IEEE Transactions on Fuzzy Systems* **12**, 661–675.

Morari, M. and Lee, J. (1999) Model predictive control: past, present and future. *Computers and Chemical Engineering* **23**, 667–682.

Muske, K.R. and Rawlings, J.B. (1993) Model predictive control with linear models. *AIChE Journal* **39**, 262–287.

Mutha, R.K., Cluett, W.R. and Penlidis, A. (1997) Nonlinear model-based predictive control of nonaffine systems. *Automatica* **33**, 907–913.

De Nicolao, G., Magni, L. and Scattolini, R. (1998) Stabilizing receding-horizon control of non-linear time-varying systems. *IEEE Transactions on Automatic Control* **43**, 1030–1036.

De Oliveira Kothare, S.L. and Morari, M. (2000) Contractive model predictive control for constrained nonlinear systems. *IEEE Transactions on Automatic Control* **45**, 1053–1071.

Oliveira, N.M.C. and Biegler, L.T. (1995) An extension of Newton-type algorithms for nonlinear process control. *Automatica* **31**, 281–286.

Qin, S.J. and Badgwell, T.A. (2003) A survey of industrial model predictive control technology. *Control Engineering Practice* **11**, 733–764.

Rawlings, J.B. and Muske, K.R. (1993) The stability of constrained receding horizon control. *IEEE Transactions on Automatic Control* **38**, 1512–1516.

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

Rouhani, R. and Mehra, R.K. (1982) Model algorithmic control (MAC); basic theoretical properties. *Automatica* **18**, 401–441.

Scattolini, R. and Bittanti, S. (1990) On the choice of the horizon in long-range predictive control – some simple criteria. *Automatica* **26**, 915–917.

Scokaert, P.O.M., Mayne, D.Q. and Rawlings, J.B. (1999) Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control* **3**, 648–654.

Scokaert, P.O.M. (1997) Infinite horizon generalized predictive control. *International Journal of Control* **66**, 161–175.

Scokaert, P.O.M., and Clarke, D.W. (1994) Stabilising properties of constrained predictive control. *Proceedings IEE, Part D*, **141**, 295–304.

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

Tatjewski, P. and Ławryńczuk, M. (2006) Soft computing in model-based predictive control. *International Journal of Applied Mathematics and Computer Science* **16**, 101–120.