Chemical Engineering Journal 145 (2008) 290–307

Contents lists available at ScienceDirect

Chemical Engineering Journal

journal homepage: www.elsevier.com/locate/cej

Modelling and nonlinear predictive control of a yeast fermentation biochemical reactor using 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

ARTICLE INFO

Article history:
Received 3 June 2008
Received in revised form 5 August 2008
Accepted 6 August 2008

Keywords:
Process control
Model predictive control
Yeast fermentation
Neural networks
Optimisation
Linearisation
Quadratic programming

ABSTRACT

This paper is concerned with using artificial neural networks for modelling and temperature control of a yeast fermentation biochemical reactor. At first, a neural model of the process is trained using available data sets generated from the fundamental model. The neural model is pruned in order to reduce its complexity and to improve its prediction ability. Next, a computationally efficient nonlinear model predictive control (MPC) algorithm with Nonlinear Prediction and Linearisation (MPC-NPL) which needs solving on-line a quadratic programming problem is developed. It is shown that the algorithm results in closed-loop control performance similar to that obtained in nonlinear MPC, which hinges on full on-line non-convex optimisation. The computational complexity of the MPC-NPL algorithm is shown, the control accuracy and the disturbance rejection ability are also demonstrated in the case of noisy measurements and disturbances affecting the process.

1. Introduction

Model Predictive Control (MPC) refers to a class of computer control algorithms that directly use an explicit dynamic model in order to predict future behaviour of the process [19,32,36]. At each sampling instant a predefined performance function is optimised on-line. As a result, a future control policy is obtained, the first element of which is actually applied to the process and the whole procedure is repeated.

MPC is recognised as the only one among advanced control techniques (defined as techniques more advanced than the PID approach) which has been exceptionally successful in numerous practical applications including chemical engineering, food processing, robotics, automotive and aerospace [31]. Because a dynamic model is used to predict future behaviour of the process, MPC algorithms have a unique ability to take into account constraints imposed on both process inputs (manipulated variables) and outputs (controlled variables) or states. Constraints are very important, they usually determine quality, economic efficiency and safety. Moreover, MPC techniques are very efficient in multivariable process control.

If it is possible, MPC algorithms based on linear models should be used because of low computational complexity [19,36]. Since properties of many technological processes are nonlinear, different nonlinear MPC techniques have been developed [9,24,31,36]. The structure of the nonlinear model and the way it is used on-line affect the accuracy, the computational burden and the reliability of nonlinear MPC. Fundamental (first-principle) models [15,21], although potentially very precise, are usually not suitable for on-line control. Such models are comprised of systems of nonlinear differential and algebraic equations which have to be solved on-line in MPC. Such an approach is usually computationally demanding as fundamental models can be very complex and may lead to numerical problems (e.g. stiffness, ill-conditioning). Moreover, in many cases development of fundamental models is difficult.

In recent years neural networks [8] have been frequently used for modelling and control of nonlinear processes [12,25,36]. It is because they have many advantages. More specifically, neural networks are universal approximators [10], hence they may be used to approximate nonlinear behaviour of technological dynamic processes. Neural networks are trained using available data sets, the necessity of developing complicated fundamental models is avoided. Unlike fundamental models, neural models have simple structures and relatively small numbers of parameters. As a result, numerical problems typical of MPC algorithms based on

☆ The work presented in this paper was supported by the Polish national budget funds for science in 2007–2009 as a research project.

* Tel.: +48 22 234 76 73; fax: +48 22 825 37 19.
E-mail address: M.Lawrynczuk@ia.pw.edu.pl.

1385-8947/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.cej.2008.08.005

Nomenclature

Notation

$0_{i\times j}$	the zeros matrix of dimensionality $i \times j$
a_l, b_l	coefficients of the linear model
$a_\mathrm{l}(\bar{\boldsymbol{x}}(k))$, $b_\mathrm{l}(\bar{\boldsymbol{x}}(k))$	coefficients of the linearised model
$\boldsymbol{A}_\mathrm{QP}$, b_QP	definition of constraints in a quadratic programming problem
A_T	heat transfer area (m^2)
A_1, A_2	exponential factors in Arhenius equation
c_{O_2}	oxygen concentration in the liquid phase (mg/l)
$c^*_{\mathrm{O}_2}$	equilibrium concentration of oxygen in the liquid phase (mg/l)
$c^*_{\mathrm{O}_2,0}$	equilibrium concentration of oxygen in distilled water (mg/l)
c_P	product (ethanol) concentration (g/l)
c_S	substrate (glucose) concentration (g/l)
$c_{\mathrm{S,in}}$	glucose concentration in the feed flow (g/l)
c_X	biomass (yeast) concentration (g/l)
$C_{\mathrm{heat,ag}}$	heat capacity of the cooling agent ($\mathrm{J\,g^{-1}\,K^{-1}}$)
$C_{\mathrm{heat,r}}$	heat capacity of the mass of the reaction ($\mathrm{J\,g^{-1}\,K^{-1}}$)
$d(k)$	estimation of the unmeasured disturbance
E_{a_1}, E_{a_2}	activation energy (J/mol)
f	neural model of the process
f_QP, $\boldsymbol{H}_\mathrm{QP}$	definition of the objective function in the a quadratic programming problem
F_ag	flow of the cooling agent (l/h)
F_e	outlet flow from the reactor (l/h)
F_i	flow of the substrate entering the reactor (l/h)
$\boldsymbol{G}(k)$	the dynamic matrix
H_i	specific ionic constant of ion i (i = Ca, Cl, CO_3, H, Mg, Na, OH)
I_i	ionic strength of ion i (i = Ca, Cl, CO_3, H, Mg, Na, OH)
$\boldsymbol{I}_{i\times j}$	the identity matrix of dimensionality $i \times j$
I_u, $I_{uf}(p)$, $I_{yp}(p)$	auxiliary coefficients in MPC-NPL
$J(k)$	the objective function of MPC
$\boldsymbol{J}^\mathrm{NPL}$	an auxiliary matrix in MPC-NPL
k	discrete time (the current sampling instant)
$(k_\mathrm{l}a)$	product of the mass-transfer coefficient for oxygen and gas-phase specific area $\mathrm{h^{-1}}$
$(k_\mathrm{l}a)_0$	product of the mass-transfer coefficient at 20 °C for O_2 and gas-phase specific area $\mathrm{h^{-1}}$
K	the number of hidden nodes in the neural network
K_{O_2}	constant for oxygen consumption (mg/l)
K_P	constant of growth inhibition by ethanol (g/l)
K_{P_1}	constant of fermentation inhibition by ethanol (g/l)
K_S	constant in the substrate term for growth (g/l)
K_{S_1}	constant in the substrate term for ethanol production (g/l)
K_T	heat transfer coefficient ($\mathrm{J\,h^{-1}\,m^{-2}\,K^{-1}}$)
$\boldsymbol{M}_1$, $\boldsymbol{M}_2$, $\boldsymbol{M}_3$	auxiliary matrices in MPC-NPL
m_i	quantity of inorganic salt i (i = $CaCO_3$, $MgCl_2$, NaCl) (g)
M_i	molecular/atomic mass of salt/ion i (g/mol)
$n_\mathrm{A}n_\mathrm{B}$	definition of the order of the dynamics
N, N_u	prediction and control horizon, respectively
r_{O_2}	rate of oxygen consumption ($\mathrm{mg\,l^{-1}\,h^{-1}}$)
R	universal gas constant ($\mathrm{J\,mol^{-1}\,K^{-1}}$)
R_SP	ratio of ethanol produced per glucose consumed for fermentation
R_SX	ratio of cell produced per glucose consumed for growth
SSE	sum of squared errors
$s_i(k)$	step-response coefficients of the linearised model
$S^1_{i,j}$, S^2_i	saliency coefficients
t	continuous time (h)
T_ag	temperature of the cooling agent in the jacket (°C)
T_in	temperature of the substrate flow entering to the reactor (°C)
$T_{\mathrm{in,ag}}$	temperature of the cooling agent entering to the jacket (°C)
T_r	temperature in the reactor (°C)
$T^\mathrm{ref}_\mathrm{r}$	reference trajectory of the temperature in the reactor (°C)
u	input of the model/proces
u^min, u^max, $\boldsymbol{u}^\mathrm{min}$, $\boldsymbol{u}^\mathrm{max}$	definition of constraints of the manipulated variable in MPC
$\boldsymbol{u}^\mathrm{NPL}(k)$	an auxiliary vector in MPC-NPL
V	volume of the mass of the reaction (l)
V_j	volume of the jacket (l)
$w^1_{i,j}$, w^2_i	weights of the neural network
$\bar{\boldsymbol{x}}(k)$	the linearisation point in MPC-NPL
x_QP	solution to the quadratic programming problem
y^min, y^max, $\boldsymbol{y}^\mathrm{min}$, $\boldsymbol{y}^\mathrm{max}$	definition of constraints of the controlled variable in MPC
y	output of the model/proces
$y^0(k+p\|k)$, $\boldsymbol{y}^0(k)$	the free trajectory
$y^\mathrm{ref}(k+p\|k)$, $\boldsymbol{y}^\mathrm{ref}(k)$	the reference trajectory
$\hat{y}(k+p\|k)$, $\hat{\boldsymbol{y}}(k)$	predicted trajectory of the controlled variable
Y_{O_2}	the amount of oxygen consumed per unit biomass produced (mg/mg)
$z_i(k)$	sums of inputs of the ith hidden node of the neural network

Greek symbols

ΔH_r	reaction heat of fermentation (kJ/mol O_2 consumed)
$\Delta u(k+p\|k)$, $\Delta\boldsymbol{u}(k)$	future increments of the manipulated variable
Δu^max, $\Delta\boldsymbol{u}^\mathrm{max}$	definition of constraints of the manipulated variable in MPC
$\boldsymbol{\varepsilon}^\mathrm{min}$, $\boldsymbol{\varepsilon}^\mathrm{max}$	slack variables in MPC optimisation problem
φ	transfer function of the hidden nodes of the neural network
μ_{O_2}	maximum specific oxygen consumption rate ($\mathrm{h^{-1}}$)
μ_P	maximum specific fermentation rate ($\mathrm{h^{-1}}$)
μ_X	maximum specific growth rate ($\mathrm{h^{-1}}$)
λ_p, $\boldsymbol{\Lambda}$	weights in MPC
ρ_ag	density of the cooling agent (g/l)
ρ^min, ρ^max	weighting coefficients in MPC optimisation problem
ρ	density of the mass of the reaction (g/l)
τ	discrete-time delay

comprehensive fundamental models are not encountered because neural models directly describe input–output relations of process variables, complicated systems of nonlinear differential and algebraic equations do not have to be solved on-line. Different versions of MPC algorithms based on neural models have been developed [2,14,17,18,25,28,29,30,36–39].

This paper is concerned with using artificial neural networks for modelling and temperature control of a yeast fermentation biochemical reactor the fundamental model of which is thoroughly described in [26]. Alcoholic fermentation is one of the most important biochemical processes. Its significance has significantly

increased recently because ethanol can be viewed as an alternative source of energy (biofuel). Because the process exhibits significantly nonlinear behaviour, the classical PID controller and the MPC algorithm based on a linear model are unable to control the process efficiently as demonstrated in [26]. In [26] a fully-fledged nonlinear MPC algorithm in which a nonlinear optimisation problem has to be solved at each sampling instant on-line is developed. As a computationally efficient alternative, an easy to implement controller based on an inverse model of the process is recommended.

In this paper a computationally efficient MPC approach to temperature control of a yeast fermentation biochemical reactor is recommended. At first, the neural model of the process is trained using available data sets generated from the fundamental model. In order to reduce the complexity of the neural model and to improve its prediction ability the neural network is pruned using the Optimal Brain Damage algorithm [13]. Next, a computationally efficient nonlinear MPC algorithm with Nonlinear Prediction and Linearisation (MPC-NPL) [16–18,36,37] developed by the author of this paper is used. The algorithm needs solving on-line a quadratic programming problem. It is shown that the algorithm results in closed-loop control performance similar to that obtained in nonlinear MPC, which hinges on full on-line non-convex optimisation. The computational complexity of the algorithm is shown, the control accuracy and the disturbance rejection ability are also demonstrated in the case of noisy measurements and disturbances affecting the process.

This paper is organised as follows. Section 2 shortly presents the general idea of MPC, describes the structure of the neural model and details the MPC-NPL algorithm. Next, in Section 3, the fundamental model of the yeast fermentation biochemical reactor is shortly presented, development of the neural model is described and simulation results of MPC algorithms based on linear and neural models are presented and discussed. Finally, Section 4 concludes the paper.

2. Computationally efficient model predictive control based on neural models

2.1. Model predictive control problem formulation

Although a number of different MPC techniques have been developed over the years, the main idea (i.e. the explicit application of a process model, optimisation of a cost function and the receding horizon approach) is always the same [19,32,36]. More specifically, at each consecutive sampling instant, k, a set of future control increments is calculated

$$\Delta \boldsymbol{u}(k) = [\Delta u(k|k) \dots \Delta u(k+N_u-1|k)]^{\mathrm{T}} \tag{1}$$

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 differences between the reference trajectory $y^{\mathrm{ref}}(k+p|k)$ and predicted values of the output $\hat{y}(k+p|k)$ over the prediction horizon $N \geq N_u$ and to penalise excessive control increments. The following quadratic cost function is typically used

$$\begin{aligned} J(k) &= \sum_{p=1}^{N} \left(y^{\mathrm{ref}}(k+p|k) - \hat{y}(k+p|k)\right)^2 \\ &\quad + \sum_{p=0}^{N_u-1} \lambda_p (\Delta u(k+p|k))^2 \end{aligned} \tag{2}$$

where $\lambda_p > 0$ are weighting factors. Only the first element of the determined sequence (1) is actually applied to the process

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

At the next sampling instant, $k+1$, the prediction is shifted one step forward, the output measurement is updated and the whole procedure is repeated.

Since constraints have to be usually taken into account, future control increments are determined from the following optimization problem

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

The general prediction equation for $p = 1, \dots, N$ is

$$\hat{y}(k+p|k) = y(k+p|k) + d(k) \tag{5}$$

where quantities $y(k+p|k)$ are calculated from a dynamic model of the process. The "DMC type" disturbance model is used in which the unmeasured disturbance $d(k)$ is assumed to be constant over the prediction horizon [36]. It is estimated from

$$d(k) = y(k) - y(k|k-1) \tag{6}$$

where $y(k)$ is measured while $y(k|k-1)$ is calculated from the dynamic model.

2.2. Neural model of the process

Predicted values of the output signal, $\hat{y}(k+p|k)$, over the prediction horizon N are calculated from (5) using a dynamic model of the process. Let the Single-Input Single-Output (SISO) process under consideration be described by the following nonlinear discrete-time equation

$$\begin{aligned} y(k) &= f\left(u(k-\tau), \dots, u(k-n_{\mathrm{B}}),\right. \\ &\quad \left. y(k-1), \dots, y(k-n_{\mathrm{A}})\right) \end{aligned} \tag{7}$$

where $f : \Re^{n_{\mathrm{A}}+n_{\mathrm{B}}-\tau+1} \to \Re$, $\tau \leq n_{\mathrm{B}}$. A feedforward Multi Layer Perceptron (MLP) neural network with one hidden layer and a linear output [8] is used as the function f in (7). The structure of the neural model is depicted in Fig. 1. Output of the model can be expressed as

$$y(k) = w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi(z_i(k)) \tag{8}$$

where $z_i(k)$ are sums of inputs of the ith hidden node, $\varphi : \Re \to \Re$ is the nonlinear transfer function (e.g. the hyperbolic tangent), K is the number of hidden nodes. Recalling the input arguments of the

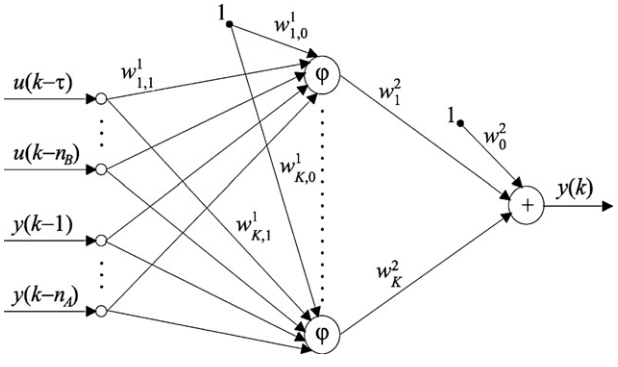

Fig. 1. The structure of the neural model.

general nonlinear model (7) one has

$$z_i(k) = w^1_{i,0} + \sum_{j=1}^{I_u} w^1_{i,j} u(k-\tau+1-j) + \sum_{j=1}^{n_A} w^1_{i,I_u+j} y(k-j) \quad (9)$$

where $I_u = n_B - \tau + 1$. Weights of the network are denoted by $w^1_{i,j}$, $i = 1, \ldots, K, j = 0, \ldots, n_A + n_B - \tau + 1$, and $w^2_i, i = 0, \ldots, K$, for the first and the second layer, respectively.

Using the general prediction Eq. (5) and the nonlinear neural model defined by (8) and (9), output predictions over the prediction horizon are calculated from

$$\hat{y}(k+p|k) = w^2_0 + \sum_{i=1}^{K} w^2_i \varphi(z_i(k+p|k)) + d(k) \quad (10)$$

Considering the prediction of the output over the horizon N for a sampling instant $k+p$ calculated at the current sampling instant k, the quantities $z_i(k+p|k)$ and consequently $\hat{y}(k+p|k)$ depend on future control signals (i.e. decision variables of the MPC algorithm), control signal values applied to the plant at previous sampling instants, future output predictions and measured values of the plant output signal. From (9) one has

$$\begin{aligned} z_i(k+p|k) = {} & w^1_{i,0} + \sum_{j=1}^{I_{uf}(p)} w^1_{i,j} u(k-\tau+1-j+p|k) \\ & + \sum_{j=I_{uf}(p)+1}^{I_u} w^1_{i,j} u(k-\tau+1-j+p) \\ & + \sum_{j=1}^{I_{yp}(p)} w^1_{i,I_u+j} \hat{y}(k-j+p|k) \\ & + \sum_{j=I_{yp}(p)+1}^{n_A} w_1(i, I_u+j) y(k-j+p) \end{aligned} \quad (11)$$

where $I_{uf}(p) = \max(\min(p-\tau+1, I_u), 0)$ and $I_{yp}(p) = \min(p-1, n_A)$. Using (6), the unmeasured disturbance is estimated from

$$d(k) = y(k) - \left(w_2(0) + \sum_{i=1}^{K} w_2(i) \varphi(z_i(k)) \right) \quad (12)$$

2.3. MPC-NPL optimisation problem

If for prediction the nonlinear neural model is used without any simplifications, predictions $\hat{y}(k+p|k)$ are nonlinear functions of future control moves (1). In such a case the nonlinear MPC optimisation problem (4) has to be solved on-line at each sampling instant. Although in theory such an approach seems to be potentially very precise, it has limited practical applicability. It is necessary to emphasise the fact that the difficulty of the nonlinear MPC optimization problem 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 optimization 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 it is required in on-line control.

Bearing in mind difficulties typical of MPC with on-line nonlinear optimisation, in this paper the MPC scheme with Nonlinear Prediction and Linearisation (MPC-NPL) [16–18,36,37]. is used. The algorithm is computationally efficient, it requires solving on-line only a quadratic programming problem. The structure of the algorithm is depicted in Fig. 2. At each sampling instant k the neural model is used on-line twice: to determine a local linearisation and a nonlinear free trajectory. 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 future input moves $\Delta\boldsymbol{u}(k)$) and the free trajectory $\boldsymbol{y}^0(k)$, which depends only on the past

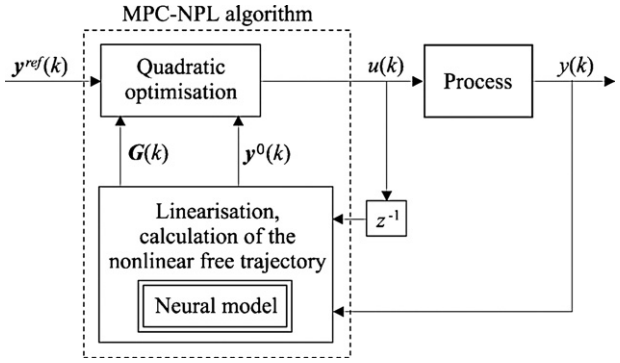

Fig. 2. The structure of the MPC algorithm with Nonlinear Prediction and Linearisation (MPC-NPL).

$$\hat{\boldsymbol{y}}(k) = \boldsymbol{G}(k)\Delta\boldsymbol{u}(k) + \boldsymbol{y}^0(k) \quad (13)$$

where $\boldsymbol{y}^0(k) = [y^0(k+1|k) \ldots y^0(k+N|k)]^T$. The dynamic matrix $\boldsymbol{G}(k)$ of dimensionality $N \times N_u$ is comprised of step-response coefficients of the linearised model

$$\boldsymbol{G}(k) = \begin{bmatrix} s_1(k) & 0 & \ldots & 0 \\ s_2(k) & s_1(k) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ s_N(k) & s_{N-1}(k) & \ldots & s_{N-N_u+1}(k) \end{bmatrix} \quad (14)$$

Both the free trajectory and the dynamic matrix are calculated on-line from the nonlinear neural model taking into account the current state of the process. The implementation of the algorithm is described in more details in the following subsection.

On the one hand, the suboptimal prediction calculated from (13) is different from the optimal one determined from the nonlinear neural model as it is done in the MPC algorithm with nonlinear optimization [17,36,37]. On the other hand, thanks to using the suboptimal prediction, the optimisation problem (4) becomes the following quadratic programming task

$$\begin{aligned} & \min_{\Delta\boldsymbol{u}(k)} \|\boldsymbol{y}^{ref}(k) - \boldsymbol{G}(k)\Delta\boldsymbol{u}(k) - \boldsymbol{y}^0(k)\|^2 + \|\Delta\boldsymbol{u}(k)\|^2_{\boldsymbol{\Lambda}} \\ & \text{subject to} \\ & \boldsymbol{u}^{min} \le \boldsymbol{J}^{NPL}\Delta\boldsymbol{u}(k) + \boldsymbol{u}^{NPL}(k) \le \boldsymbol{u}^{max} \\ & -\Delta\boldsymbol{u}^{max} \le \Delta\boldsymbol{u}(k) \le \Delta\boldsymbol{u}^{max} \\ & \boldsymbol{y}^{min} \le \boldsymbol{G}(k)\Delta\boldsymbol{u}(k) + \boldsymbol{y}^0(k) \le \boldsymbol{y}^{max} \end{aligned} \quad (15)$$

where

$$\boldsymbol{y}^{ref} = [y^{ref}(k+1|k) \ldots y^{ref}(k+N|k)]^T \quad (16)$$

$$\boldsymbol{y}^{min} = [y^{min} \ldots y^{min}]^T \quad (17)$$

$$\boldsymbol{y}^{max} = [y^{max} \ldots y^{max}]^T \quad (18)$$

are vectors of length N,

$$\boldsymbol{u}^{min} = [u^{min} \ldots u^{min}]^T \quad (19)$$

$$\boldsymbol{u}^{max} = [u^{max} \ldots u^{max}]^T \quad (20)$$

$$\Delta\boldsymbol{u}^{max} = [\Delta u^{max} \ldots \Delta u^{max}]^T \quad (21)$$

$$\boldsymbol{u}^{NPL}(k) = [u(k-1) \ldots u(k-1)]^T \quad (22)$$

are vectors of length N_u, $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_0, \ldots, \lambda_{N_u-1})$. $\boldsymbol{J}^{\mathrm{NPL}}$ is the all ones lower triangular matrix of dimensionality $N_u \times N_u$.

If output constraints have to be taken into account, the MPC-NPL optimisation task (15) may be affected by the infeasibility problem, i.e. the admissible set of the optimization problem may be empty. In order to cope with such a situation, output constraints have to be softened by means of slack variables [19,36]. The MPC-NPL quadratic programming problem becomes

$$\begin{aligned} &\min_{\Delta\boldsymbol{u}(k),\boldsymbol{\varepsilon}^{\min},\boldsymbol{\varepsilon}^{\max}} \left\{ \begin{array}{l} \|\boldsymbol{y}^{\mathrm{ref}}(k)-\boldsymbol{y}^0(k)-\boldsymbol{G}(k)\Delta\boldsymbol{u}(k)\|^2+\|\Delta\boldsymbol{u}(k)\|^2_{\boldsymbol{\Lambda}} \\ \\ +\rho^{\min}\|\boldsymbol{\varepsilon}^{\min}\|^2+\rho^{\max}\|\boldsymbol{\varepsilon}^{\max}\|^2 \end{array} \right\} \\ &\text{subject to} \\ &\boldsymbol{u}^{\min} \le \boldsymbol{J}\Delta\boldsymbol{u}(k) + \boldsymbol{u}^{k-1}(k) \le \boldsymbol{u}^{\max} \\ &-\Delta\boldsymbol{u}^{\max} \le \Delta\boldsymbol{u}(k) \le \Delta\boldsymbol{u}^{\max} \\ &\boldsymbol{y}^{\min} - \boldsymbol{\varepsilon}^{\min} \le \boldsymbol{y}^0(k) + \boldsymbol{G}(k)\Delta\boldsymbol{u}(k) \le \boldsymbol{y}^{\max} + \boldsymbol{\varepsilon}^{\max} \\ &\boldsymbol{\varepsilon}^{\min} \ge 0, \quad \boldsymbol{\varepsilon}^{\max} \ge 0 \end{aligned} \tag{23}$$

A quadratic penalty for constraint violations is used in the MPC optimisation problem (23), $\boldsymbol{\varepsilon}^{\min}$ and $\boldsymbol{\varepsilon}^{\max}$ are vectors of length N comprising the slack variables and $\rho^{\min}$, $\rho^{\max} > 0$ are weights.

Let $x_{\mathrm{QP}} = [\Delta\boldsymbol{u}^{\mathrm{T}}(k)(\boldsymbol{\varepsilon}^{\min})^{\mathrm{T}}(\boldsymbol{\varepsilon}^{\max})^{\mathrm{T}}]^{\mathrm{T}}$ be a vector containing all decision variables of the MPC-NPL algorithm. The optimisation problem (23) can be easily rewritten in a standard quadratic programming (QP) form

$$\begin{aligned} &\min_{x_{\mathrm{QP}}}\{\tfrac{1}{2}x_{\mathrm{QP}}^{\mathrm{T}}\boldsymbol{H}_{\mathrm{QP}}x_{\mathrm{QP}} + f_{\mathrm{QP}}^{\mathrm{T}}x_{\mathrm{QP}}\} \\ &\text{subject to} \\ &\boldsymbol{A}_{\mathrm{QP}}x_{\mathrm{QP}} \le b_{\mathrm{QP}} \end{aligned} \tag{24}$$

where

$$\Delta\boldsymbol{u}(k) = \boldsymbol{M}_1 x_{\mathrm{QP}}, \quad \boldsymbol{M}_1 = [\boldsymbol{I}_{N_u\times N_u} 0_{N_u\times 2N}] \tag{25}$$

$$\boldsymbol{\varepsilon}^{\min} = \boldsymbol{M}_2 x_{\mathrm{QP}}, \quad \boldsymbol{M}_2 = [0_{N\times N_u} \boldsymbol{I}_{N\times N} 0_{N\times N}] \tag{26}$$

$$\boldsymbol{\varepsilon}^{\max} = \boldsymbol{M}_3 x_{\mathrm{QP}}, \quad \boldsymbol{M}_3 = [0_{N\times N_u} 0_{N\times N} \boldsymbol{I}_{N\times N}] \tag{27}$$

The cost function in the QP problem (24) is defined by

$$\begin{aligned} \boldsymbol{H}_{\mathrm{QP}} &= 2(\boldsymbol{M}_1^{\mathrm{T}}\boldsymbol{G}^{\mathrm{T}}(k)\boldsymbol{G}(k)\boldsymbol{M}_1 + \boldsymbol{M}_1^{\mathrm{T}}\boldsymbol{\Lambda}\boldsymbol{M}_1) \\ &\quad +2\rho^{\min}\boldsymbol{M}_2^{\mathrm{T}}\boldsymbol{M}_2 + 2\rho^{\max}\boldsymbol{M}_3^{\mathrm{T}}\boldsymbol{M}_3 \end{aligned} \tag{28}$$

$$f_{\mathrm{QP}} = -2\boldsymbol{M}_1^{\mathrm{T}}\boldsymbol{G}^{\mathrm{T}}(k)(\boldsymbol{y}^{\mathrm{ref}}(k) - \boldsymbol{y}^0(k)) \tag{29}$$

whereas constraints by

$$\boldsymbol{A}_{\mathrm{QP}} = \begin{bmatrix} -\boldsymbol{J}\boldsymbol{M}_1 \\ \boldsymbol{J}\boldsymbol{M}_1 \\ -\boldsymbol{G}(k)\boldsymbol{M}_1 - \boldsymbol{M}_2 \\ \boldsymbol{G}(k)\boldsymbol{M}_1 - \boldsymbol{M}_3 \\ -\boldsymbol{M}_1 \\ \boldsymbol{M}_1 \\ -\boldsymbol{M}_2 \\ -\boldsymbol{M}_3 \end{bmatrix}, \quad f_{\mathrm{QP}} = \begin{bmatrix} -\boldsymbol{u}^{\min} + \boldsymbol{u}^{k-1}(k) \\ \boldsymbol{u}^{\max} - \boldsymbol{u}^{k-1}(k) \\ -\boldsymbol{y}^{\min} + \boldsymbol{y}^0(k) \\ \boldsymbol{y}^{\max} - \boldsymbol{y}^0(k) \\ \Delta\boldsymbol{u}^{\max} \\ \Delta\boldsymbol{u}^{\max} \\ 0_{N\times 1} \\ 0_{N\times 1} \end{bmatrix} \tag{30}$$

As shown in Fig. 2, in the MPC-NPL algorithm at each sampling instant k the following steps are repeated:

1. Linearisation of the neural model: obtain step response coefficients $s_1(k), \ldots, s_N(k)$ comprising the dynamic matrix $\boldsymbol{G}(k)$.
2. Calculate the nonlinear free trajectory $\boldsymbol{y}^0(k)$ using the neural model.
3. Solve the quadratic programming problem (24) 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.

It has to be pointed out that the general idea of computing the optimal control law for the current linear approximation of the nonlinear model is known from the literature [9,24]. If the constraints are not taken into account, such an approach leads to the solution of a state-dependent Riccati equation, it is known as the "frozen Riccati equation technique". An important feature of the unconstrained case is the fact that its stability properties [11,27] can be analysed in significantly easier way in comparison with the constrained one [20]. Because of the same reason and the simplicity of implementation, unconstrained MPC algorithms which use the local linearisation approach are frequently considered in the literature [22,36].

In practice stability of the MPC-NPL algorithm can be achieved by proper tuning of the prediction horizon and weighting coefficients λ_p. Furthermore, the algorithm can be combined with the stabilising dual-mode approach [23] as detailed in [16].

2.4. On-line linearisation of the neural model and calculation of the nonlinear free trajectory

Defining the linearisation point as a vector $\bar{\boldsymbol{x}}(k)$ composed of past input and output signal values corresponding to the arguments of the nonlinear model (7)

$$\bar{\boldsymbol{x}}(k) = [\bar{u}(k-\tau)\ldots\bar{u}(k-n_{\mathrm{B}})\bar{y}(k-1)\ldots\bar{y}(k-n_{\mathrm{A}})]^{\mathrm{T}} \tag{31}$$

and using Taylor series expansion at this point, the linear approximation of the nonlinear model, obtained at a sampling instant k, can be expressed as

$$\begin{aligned} y(k) &= f(\bar{\boldsymbol{x}}(k)) + \sum_{l=1}^{n_{\mathrm{B}}} b_l(\bar{\boldsymbol{x}}(k))(u(k-l) - \bar{u}(k-l+1)) \\ &\quad - \sum_{l=1}^{n_{\mathrm{A}}} a_l(\bar{\boldsymbol{x}}(k))(y(k-l) - \bar{y}(k-l+1)) \end{aligned} \tag{32}$$

where $a_l(\bar{\boldsymbol{x}}(k)) = -(\partial f(\bar{\boldsymbol{x}}(k)))/(\partial y(k-l))$ and $b_l(\bar{\boldsymbol{x}}(k)) = (\partial f(\bar{\boldsymbol{x}}(k)))/(\partial u(k-l))$ are coefficients of the linearised model. Taking into account the structure of the neural model shown in Fig. 1 and defined by (8) and (9) one obtains

$$a_l(\bar{\boldsymbol{x}}(k)) = -\sum_{i=1}^{K} w_i^2 \frac{\mathrm{d}\varphi(z_i(\bar{\boldsymbol{x}}(k)))}{\mathrm{d}z_i(\bar{\boldsymbol{x}}(k))} w_{i,I_u+l}^1 \tag{33}$$

where $l = 1, \ldots, n_{\mathrm{A}}$, and

$$b_l(\bar{\boldsymbol{x}}(k)) = \begin{cases} 0 & \text{if } l = 1, \ldots, \tau-1 \\ \displaystyle\sum_{i=1}^{K} w_i^2 \frac{\mathrm{d}\varphi(z_i(\bar{\boldsymbol{x}}(k)))}{\mathrm{d}z_i(\bar{\boldsymbol{x}}(k))} w_{i,l-\tau+1}^1 & \text{if } l = \tau, \ldots, n_{\mathrm{B}} \end{cases} \tag{34}$$

If hyperbolic tangent is used as the nonlinear transfer function φ in the hidden layer of the neural model, one has $(\mathrm{d}\varphi(z_i(\bar{\boldsymbol{x}}(k))))/(\mathrm{d}z_i(\bar{\boldsymbol{x}}(k))) = 1 - \tanh^2(z_i(\bar{\boldsymbol{x}}(k)))$.

Step-response coefficients of the linearised model comprising the dynamic matrix $\boldsymbol{G}(k)$ given by (14) are determined from

$$\begin{aligned} s_j(k) &= \sum_{i=1}^{\min(j,n_{\mathrm{B}})} b_i(\bar{\boldsymbol{x}}(k)) \\ &\quad - \sum_{i=1}^{\min(j-1,n_{\mathrm{A}})} a_i(\bar{\boldsymbol{x}})(k) s_{j-i}(k) \end{aligned} \tag{35}$$

The nonlinear free trajectory $y^0(k+p|k)$ over the prediction horizon, i.e. for $p = 1, \ldots, N$, is calculated recursively from the general prediction Eq. (5) assuming only the influence of the past. The

"DMC type" disturbance model (12) is also used. Using (10), one has

$$y^0(k+p|k) = w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi(z_i^0(k+p|k)) + d(k) \tag{36}$$

Quantities $z_i^0(k+p|k)$ are determined from (11) assuming no changes in control signals from a sampling instant k onwards and replacing the output predictions by corresponding values of the free trajectory, i.e. $u(k+p|k) := u(k-1)$ for $p \geq 0$, $\hat{y}(k+p|k) := y^0(k+p|k)$ for $p \geq 1$. One has

$$\begin{aligned} z_i(k+p|k) &= w_{i,0}^1 + \sum_{j=1}^{I_{uf}(p)} w_{i,j}^1 u(k-1) \\ &+ \sum_{j=I_{uf}(p)+1}^{I_u} w_{i,j}^1 u(k-\tau+1-j+p) \\ &+ \sum_{j=1}^{I_{yp}(p)} w_{i,I_u+j}^1 y^0(k-j+p|k) \\ &+ \sum_{j=I_{yp}(p)+1}^{n_A} w_{i,I_u+j}^1 y(k-j+p) \end{aligned} \tag{37}$$

3. Experiments

3.1. Biochemical reactor

The considered yeast fermentation biochemical reactor is shown in Fig. 3. The reactor is modelled as a continuous stirred tank with constant substrate feed flow, the outlet flow from the reactor containing the product, the substrate and the biomass is also constant. The reactor contains three distinct components: the biomass, which is a suspension of yeast fed into the system and evacuated continuously, the substrate, which is the solution of glucose feeding the micro-organism (charomyces cerevisiae) and the product (ethanol), which is evacuated together with other components. Together with yeast, inorganic salts are added. It is necessary for the formation of coenzymes. Inorganic salts have also a strong influence on the equilibrium concentration of oxygen in the liquid phase. Because a low dilution rate is necessary, the process has very slow dynamic properties.

The comprehensive fundamental model of the process is described in [26], here the model is given in a compact form for completeness of presentation. The model contains the detailed kinetic model, it takes into account the heat transfer, the dependence of kinetic parameters on temperature, the mass transfer of oxygen, the influence of the temperature and the ionic strength on the mass transfer coefficient. Let state variables be defined as follows: V—the volume of the mass of the reaction (l), c_X—the biomass (yeast) concentration (g/l), c_P—the product (ethanol) concentration (g/l), c_S—the substrate (glucose) concentration (g/l), c_{O_2}—oxygen concentration in the liquid phase (mg/l), T_r—the temperature of the reactor (°C), T_{ag}—the temperature of the cooling agent in the jacket (°C). The reactor is described by the following continuous-time fundamental model containing 7 nonlinear ordinary differential equations

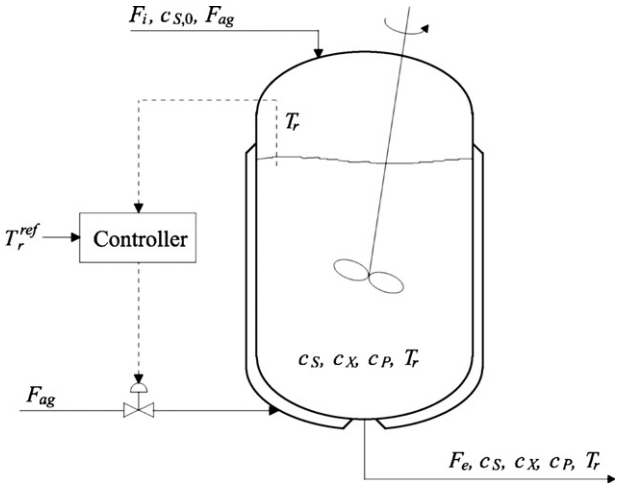

Fig. 3. The continuous yeast fermentation reactor control system structure.

$$\frac{dV}{dt} = F_i - F_e \tag{38}$$

$$\frac{dc_X}{dt} = \mu_X C_X \frac{c_S}{K_S + c_S} \exp(-K_P c_P) - \frac{F_e}{V} c_X \tag{39}$$

$$\frac{dc_P}{dt} = \mu_P C_X \frac{c_S}{K_{S_1} + c_S} \exp(-K_{P_1} c_P) - \frac{F_e}{V} c_P \tag{40}$$

$$\begin{aligned} \frac{dc_S}{dt} &= -\frac{1}{R_{SX}} \mu_X c_X \frac{c_S}{K_S + c_S} \exp(-K_P c_P) \\ &- \frac{1}{R_{SP}} \mu_P c_X \frac{c_S}{K_{S_1} + c_S} \exp(-K_{P_1} c_P) \\ &+ \frac{F_i}{V} c_{S,in} - \frac{F_e}{V} c_S \end{aligned} \tag{41}$$

$$\frac{dc_{O_2}}{dt} = (k_l a)(c_{O_2}^* - c_{O_2}) - r_{O_2} - \frac{F_e}{V} c_{O_2} \tag{42}$$

$$\begin{aligned} \frac{dT_r}{dt} &= \frac{F_i}{V}(T_{in} + 273) - \frac{F_e}{V}(T_r + 273) \\ &+ \frac{r_{O_2} \Delta H_r}{32 \rho_r C_{heat,r}} - \frac{K_T A_T (T_r - T_{ag})}{V \rho_r C_{heat,r}} \end{aligned} \tag{43}$$

$$\frac{dT_{ag}}{dt} = \frac{F_{ag}}{V_j}(T_{in,ag} - T_{ag}) + \frac{K_T A_T (T_r - T_{ag})}{V_j \rho_{ag} C_{heat,ag}} \tag{44}$$

The equilibrium concentration of oxygen in the liquid phase is

$$c_{O_2}^* = (14.6 - 0.3943 T_r + 0.007714 T_r^2 - 0.0000646 T_r^3) 10^{-\sum H_i I_i} \tag{45}$$

where the global effect of ionic strengths is

$$\begin{aligned} \sum H_i I_i &= 0.5 H_{Na} \frac{m_{NaCl}}{M_{NaCl}} \frac{M_{Na}}{V} + 2 H_{Ca} \frac{m_{CaCO_3}}{M_{CaCO_3}} \frac{M_{Ca}}{V} \\ &+ 2 H_{Mg} \frac{m_{MgCl_2}}{M_{MgCl_2}} \frac{M_{Mg}}{V} + 0.5 H_{Cl} \left(\frac{m_{NaCl}}{M_{NaCl}} + 2 \frac{m_{MgCl_2}}{M_{MgCl_2}} \right) \frac{M_{Cl}}{V} \\ &+ 2 H_{CO_3} \frac{m_{CaCO_3}}{M_{CaCO_3}} \frac{M_{CO_3}}{V} + 0.5 H_H 10^{-pH} + 0.5 H_{OH} 10^{-(14-pH)} \end{aligned} \tag{46}$$

The mass transfer coefficient for oxygen is

$$(k_l a) = (k_l a)_0 1.024^{T_r - 20} \tag{47}$$

The rate of oxygen consumption is

$$r_{O_2} = \mu_{O_2} \frac{1}{Y_{O_2}} c_X \frac{c_{O_2}}{K_{O_2} + c_{O_2}} \tag{48}$$

The maximum specific growth rate is

$$\mu_X = A_1 \exp\left(-\frac{E_{a_1}}{R(T_r + 273)} \right) - A_2 \exp\left(-\frac{E_{a_2}}{R(T_r + 273)} \right) \tag{49}$$

Parameters of the fundamental model are given in Table 1 while nominal operating conditions of the process are given in Table 2.

Table 1
Parameters of the fundamental model

$A_1 = 9.5 \times 10^8$	$(k_l a)_0 = 38\,h^{-1}$	$M_{Mg} = 24\,g/mol$
$A_2 = 2.55 \times 10^{33}$	$K_{O_2} = 8.86\,mg/l$	$M_{MgCl_2} = 95\,g/mol$
$A_T = 1\,m^2$	$K_P = 0.139\,g/l$	$M_{Na} = 23\,g/mol$
$C_{heat,ag} = 4.18\,J\,g^{-1}\,K^{-1}$	$K_{P_1} = 0.07\,g/l$	$M_{NaCl} = 58.5\,g/mol$
$C_{heat,r} = 4.18\,J\,g^{-1}\,K^{-1}$	$K_S = 1.03\,g/l$	$R = 8.31\,J\,mol^{-1}\,K^{-1}$
$E_{a_1} = 55,000\,J/mol$	$K_{S_1} = 1.68\,g/l$	$R_{SP} = 0.435$
$E_{a_2} = 220,000\,J/mol$	$K_T = 3.6 \times 10^5\,J\,h^{-1}\,m^{-2}\,K^{-1}$	$R_{SX} = 0.607$
$H_{Ca} = -0.303$	$m_{CaCO_3} = 100\,g$	$V_j = 50\,l$
$H_{Cl} = 0.844$	$m_{MgCl_2} = 100\,g$	$Y_{O_2} = 0.97\,mg/mg$
$H_{CO_3} = 0.485$	$m_{NaCl} = 500\,g$	$\Delta H_r = 518\,kJ/molØ_2$
$H_H = -0.774$	$M_{Ca} = 40\,g/mol$	$\mu_{O_2} = 0.5\,h^{-1}$
$H_{Mg} = -0.314$	$M_{CaCO_3} = 90\,g/mol$	$\mu_P = 1.79\,h^{-1}$
$H_{Na} = -0.550$	$M_{Cl} = 35.5\,g/mol$	$\rho_{ag} = 1000\,g/l$
$H_{OH} = 0.941$	$M_{CO_3} = 60\,g/mol$	$\rho_r = 1080\,g/l$

Table 2
Nominal operating conditions of the process

$c_{O_2} = 3.106953\,mg/l$	pH 6
$c_P = 12.515241\,g/l$	$T_{ag} = 27.053939\,^\circ C$
$c_S = 29.738924\,g/l$	$T_{in} = 25\,^\circ C$
$c_{S,in} = 60\,g/l$	$T_{in,ag} = 15\,^\circ C$
$c_X = 0.904677\,g/l$	$T_r = 29.573212\,^\circ C$
$F_{ag} = 18\,l/h$	$V = 1000\,l$
$F_i = F_e = 51\,l/h$	

As shown in Fig. 3, the temperature of the reactor T_r is controlled by manipulating the flow of the cooling agent F_{ag}. It means that from the perspective of a control engineer, the considered process has one input (F_{ag}) and one output (T_r). The process exhibits nonlinear behaviour. Both steady-state and dynamic properties of the yeast fermentation reactor are nonlinear. The steady-state characteristic $T_r(F_{ag})$ is shown in Figs. 4 and 5 depict open-loop step-responses.

As discussed in [26], two main disturbances can be considered: changes in the substrate concentration c_S and in the temperature of the substrate flow entering the reactor T_{in}, but only the second one has a significant effect on the process and should be considered as an important disturbance. The steady-state characteristic $T_r(F_{ag}, T_{in})$, which shows the dependence of the flow of the cooling agent and the temperature of input flow on the temperature of the reactor is depicted in Fig. 6. Fig. 7 shows open-loop step-responses of the reactor to a changes in the temperature of the substrate flow.

3.2. Yeast fermentation reactor modelling for control

For identification the fundamental model (38)–(49) is used as the real process, it is simulated open-loop in order to obtain two

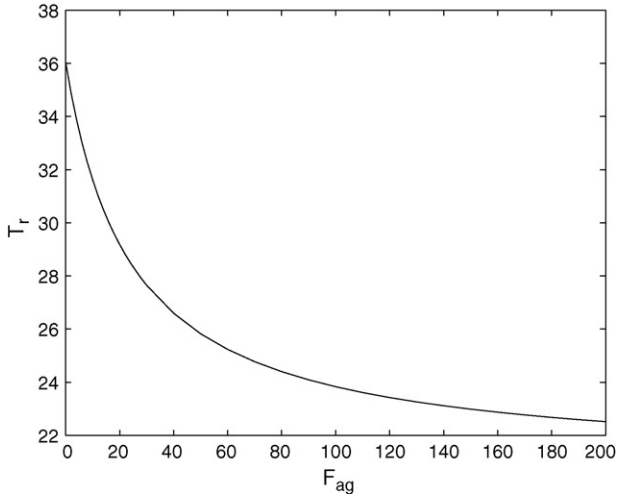

Fig. 4. The steady-state characteristic $T_r(F_{ag})$ of the reactor.

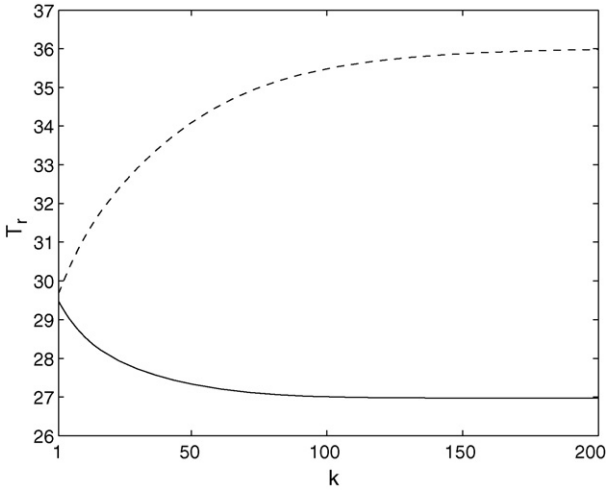

Fig. 5. Open-loop step-responses of the reactor caused by increasing (*solid line*) and decreasing (*dashed line*) the flow of the cooling agent F_{ag} by 18 l/h at $k = 1$.

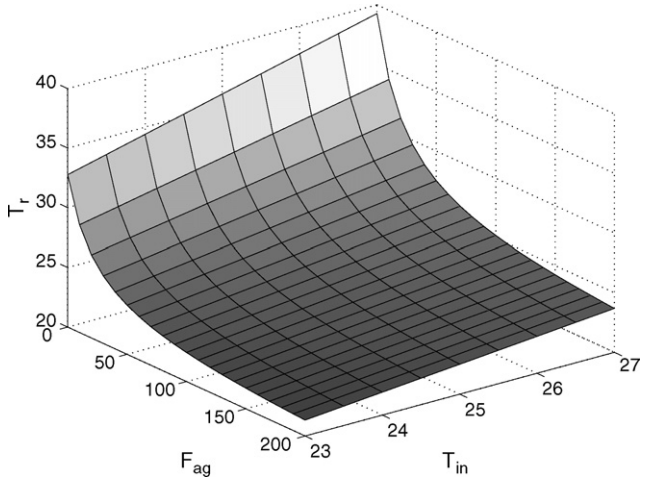

Fig. 6. The steady-state characteristic $T_r(F_{ag}, T_{in})$ of the reactor.

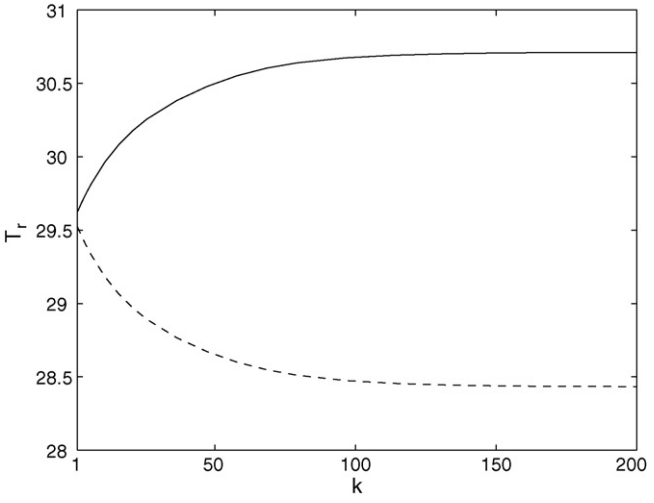

Fig. 7. Open-loop step-responses of the reactor caused by increasing (*solid line*) and decreasing (*dashed line*) the temperature T_{in} of the substrate flow entering the reactor by 1 ° C at $k = 1$.

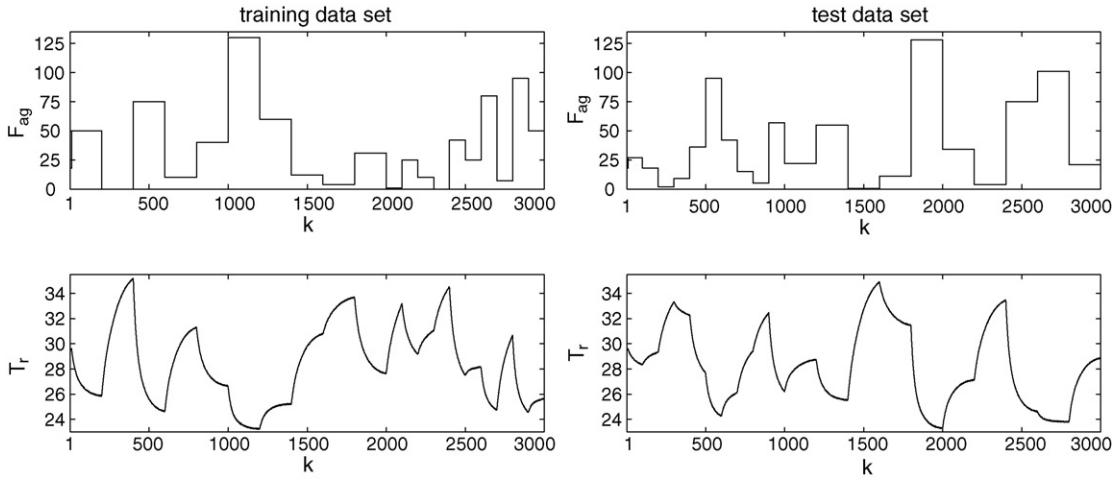

Fig. 8. Training and test data sets.

sets of data, namely training and test data sets depicted in Fig. 8. Both sets contain 3000 samples. Because the dynamic properties of the process are very slow, the sampling time is 30 min. The output signal contains small measurement noise. During simulations the system of differential equations comprising the fundamental model is solved. The standard method (Runge-Kutta 45, the solver ODE45 in Matlab) is inefficient because the problem is stiff. More specifically, it requires a very big number of iterations (steps). For example, in order to obtain open-loop step-responses of the reactor shown in Fig. 5 the solver RK45 needs as many as 24128 steps when the flow of the cooling agent F_{ag} is decreased and 16460 steps when the flow in increased. That is why the specialised solver ODE23S [33] for stiff differential equations is used (it is based on a modified Rosenbrock formula of order two). For the same open-loop step-responses this solver requires as few as 24 and 20 steps, respectively.

During model identification the following Sum of Squared Errors (SSE) performance function is minimized

$$\mathrm{SSE} = \sum_{k \in \text{ data set}} (y(k|k-1) - y(k))^2 \tag{50}$$

where $y(k|k-1)$ denotes the output of the model for the sampling instant k calculated using signals up to the sampling instant $k-1$, $y(k)$ is the real value of the process output variable collected during the identification experiment.

Second-order dynamic neural models

$$y(k) = f(u(k-1), u(k-2), y(k-1), y(k-2)) \tag{51}$$

are considered. Because input and output process variables have different orders of magnitude, they are scaled as $u = 0.01(F_{ag} - F_{ag,nom})$, $y = 0.1(T_r - T_{r,nom})$ where $F_{ag,nom} = 18$ l/h, $T_{r,nom} = 29.573212\,^{\circ}$C correspond to the nominal operating conditions of the process (Table 2). All compared neural models have the same input arguments determined by $\tau = 1$, $n_A = n_B = 2$, the difference is in the number of hidden nodes K. Neural networks containing $K = 3, 4, 5, 6, 7, 8$ hidden nodes are considered. Second-order dynamic models are used because first-order ones turn out to be insufficiently accurate while third-order structures do not improve the accuracy significantly. The hyperbolic tangent transfer function is used in the hidden layer.

Table 3 compares the accuracy of neural models in terms of Sum of Squared Errors for training and test data sets, total numbers of parameters (weights) are also given. The influence of the number of hidden nodes K on the accuracy of neural models for training and test data sets is also depicted in Fig. 9. Different training algorithms have been tested: the rudimentary backpropagation scheme (i.e. the steepest descent), the conjugate gradient methods (Polak-Ribiere, Fletcher-Reeves) and the quasi-Newton algorithms (DFP, BFGS) [1]. Finally, all neural models are trained using the BFGS algorithm, which outperforms all the aforementioned competitors in terms of learning time. Such an observation is not surprising, since neural network training task is in fact an unconstrained minimisation problem with the SSE performance index as the objective function. For each neural model structure the identification experiment is repeated 10 times, weights of neural networks are initialised randomly. The results presented are the best obtained.

Table 3
The influence of the number of hidden nodes K on the complexity and the accuracy of neural models for training and test data sets

K	No. of weights	$\mathrm{SSE}_{\mathrm{training}}$	$\mathrm{SSE}_{\mathrm{test}}$
3	19	1.097013×10^{-1}	2.991501×10^{-1}
4	25	7.638250×10^{-2}	1.235549×10^{-1}
5	31	7.073799×10^{-2}	9.913449×10^{-2}
6	37	6.698869×10^{-2}	9.858591×10^{-2}
7	43	6.031582×10^{-2}	1.473690×10^{-1}
8	49	5.617879×10^{-2}	1.719373×10^{-1}

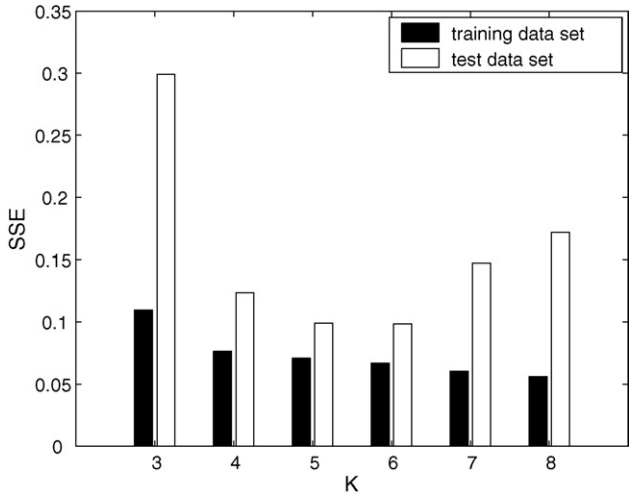

Fig. 9. The influence of the number of hidden nodes K on the accuracy of neural models for training and test data sets.

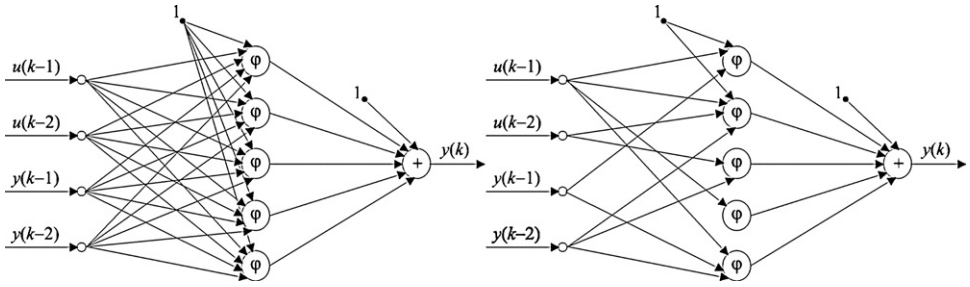

Fig. 10. The structure of the neural model before (*left*) and after pruning (*right*).

On the one hand, increasing the number of hidden nodes leads to reducing the SSE performance index for the training data set. On the other hand, it is a well known fact that neural models with too many parameters (weights) have poor generalisation abilities (overfitting). It is easily observed in the case of the considered neural models of the yeast fermentation reactor. For the test data set the value of the SSE performance index rapidly increases when $K \geq 7$. The model which is next used in MPC algorithms should have good prediction accuracy and be relatively uncomplicated. Considering neural models whose parameters are summarised in Table 3, the model with $K = 5$ hidden nodes is chosen as a reasonable compromise between the accuracy and the complexity. Of course, one can also choose the model containing $K = 6$nodes, but its SSE is only 5.30% (the training data set) and only 0.55% (the test data set) smaller whereas it has 6 weights more, which means that its complexity is 19.35% bigger.

In order to further improve the accuracy of the neural model and to reduce the number of weights the neural network can be pruned. In [26] a version of the Optimal Brain Surgeon (OBS) pruning algorithm [7] detailed in [6] is used. In this paper the Optimal Brain Damage (OBD) algorithm [13] is used. In comparison with the OBS algorithm its implementation is simpler, but it results in similar reduction of the model complexity. For pruning, the OBD algorithm uses information of second-order derivatives of the SSE performance function which is minimised during training. The second-order Taylor expansion of SSE about its minimum is used. The objective is to find a set of weights whose removing is likely to result in the least change in SSE. In order to achieve reasonable low computational complexity, only diagonal terms of the second-order Taylor expansion are included into the definition of the saliency of parameters. This corresponds to the assumption that the Hessian matrix is diagonal. A weight of the network is removed when its saliency is small. For the first layer of the network the saliency is defined as

$$S_{i,j}^1 = \frac{1}{2}\frac{\partial^2 \mathrm{SSE}}{\partial (w_{i,j}^1)^2}(w_{i,j}^1)^2 \tag{52}$$

where $i = 1, \ldots, K, j = 0, \ldots, n_\mathrm{A} + n_\mathrm{B} - \tau + 1$. For the second layer the saliency is

$$S_i^2 = \frac{1}{2}\frac{\partial^2 \mathrm{SSE}}{\partial (w_i^2)^2}(w_i^2)^2 \tag{53}$$

where $i = 0, \ldots, K$. $(\partial^2 \mathrm{SSE})/(\partial (w_{i,j}^1)^2)$ and $(\partial^2 \mathrm{SSE})/(\partial (w_i^2)^2)$ denote the second-order derivatives of the SSE performance function with

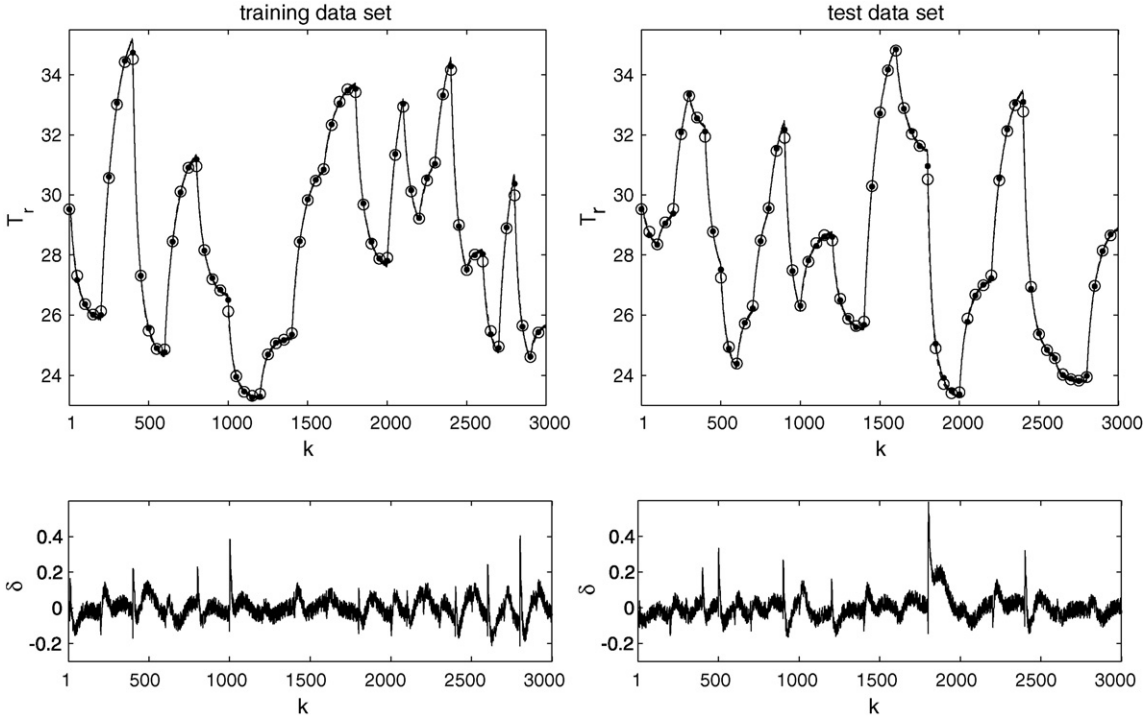

Fig. 11. *Top*: the process (*solid line with points*) vs. the neural model (*dashed line with circles*) for training and test data sets; *bottom*: prediction errors.

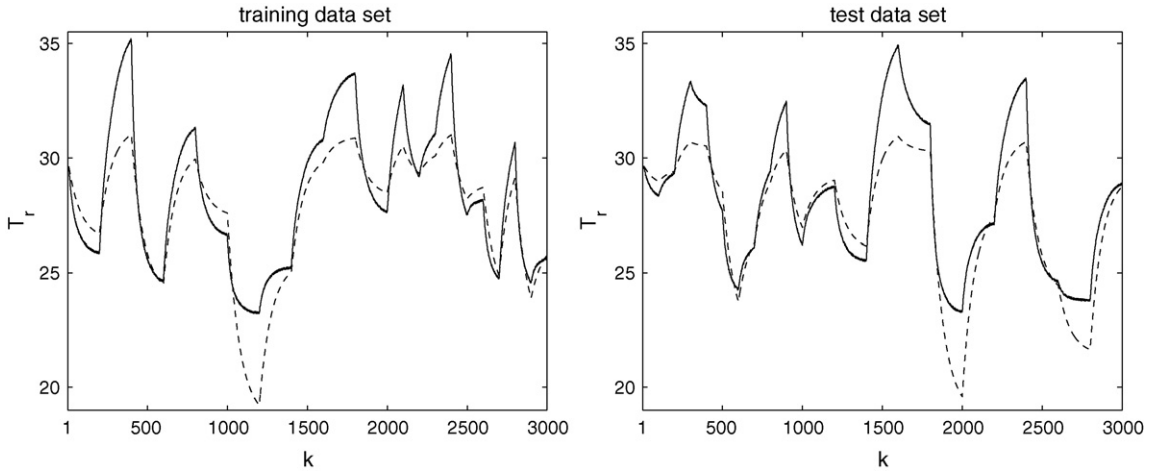

Fig. 12. The process (*solid line*) vs. the linear model (*dashed line*) for training and test data sets.

respect to weights of the first and the second layer, respectively. The OBD pruning algorithm can be summarized as follows:

1. Train the rudimentary neural network.
2. Calculate saliency coefficients $S_{i,j}^1$ and S_i^2.
3. Eliminate one weight the saliency of which is the smallest.
4. Retrain the neural network.
5. Stop if pruning results in increasing the SSE performance index.
6. Go to step 2.

Initially, the neural network contains 5 hidden nodes, it has has 31 weights. As a result of the OBD pruning algorithm, 12 weights are removed, which means that the complexity of the rudimentary neural network is reduced by 38%. The structure of the neural network before and after pruning is shown in Fig. 10. The SSE performance index of the pruned neural network for the training data set is 6.827791×10^{-2}, which means that the accuracy of the model is very close to that of the rudimentary (i.e. not pruned) model containing $K = 6$ hidden nodes (Table 3). Additionally, for the test data set the SSE performance index is 9.219128×10^{-2}. Fig. 11 shows the output of the process and the output of the neural model for both training and test data sets. Because the accuracy of the neural model is very high, prediction errors are also shown. All things considered, thanks to pruning the complexity of the neural model is considerably reduced, is has good generalisation abilities.

It is an interesting question if a linear model with constant parameters would lead to a similar modelling accuracy. The question is important because MPC algorithms which base on linear models can be easily implemented and their computational complexity is low. The linear second-order dynamic model

$$y(k) = b_1 u(k-1) + b_2 u(k-2) - a_1 y(k-1) - a_2 y(k-2) \tag{54}$$

is then found. The linear model has the same input arguments as the neural one (54). Unfortunately, because the process exhibits significantly nonlinear behaviour as shown in Figs. 4 and 5, the accuracy of the linear model is low. Fig. 12 compares the output of the process and the output of the linear model for both training and test data sets. For the training data set SSE = 73.205301, for the test data set SSE = 69.846697.

Neural networks are universal approximators [10], which means that a multilayer perceptron with one hidden layer can approximate any smooth function to an arbitrary degree of accuracy. On the other hand, the question still remains whether the process can be represented by a a simpler model than the neural one. Although different model structures can be used [5], an interesting idea is to find the state-dependent parameter representations,

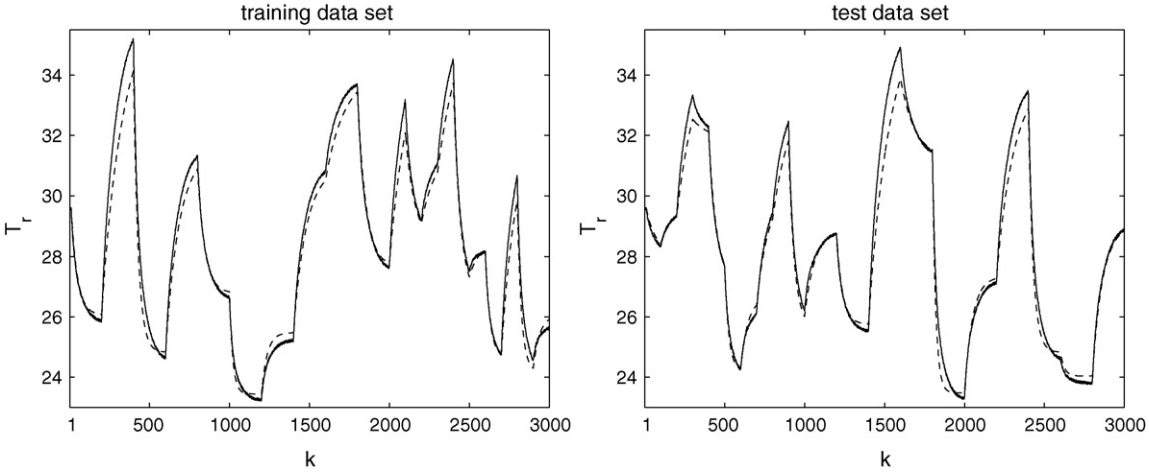

Fig. 13. The process (*solid line*) vs. the nonlinear polynomial model (*dashed line*) for training and test data sets.

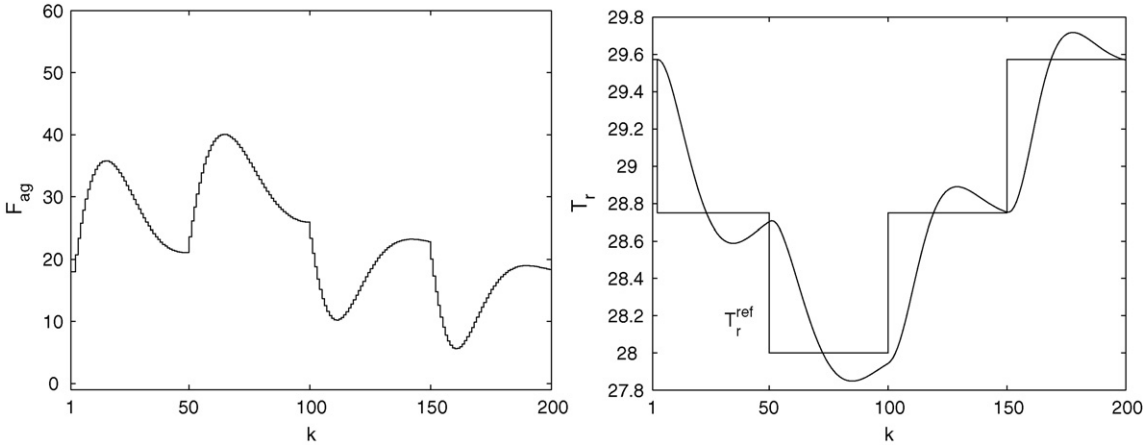

Fig. 14. Experiment 1 (the reference trajectory 1): the GPC algorithm based on the linear model.

i.e. a model whose parameters depend on the state of the process. For example, in [35] a nonlinear multivariable distillation column which has a state-dependent dynamic response similar to that of the discussed biochemical reactor is modelled by a simple linear model, whose parameters depend affinely on process outputs. The state-dependent parameter modelling is a well-established technique. Considering the steady-state characteristic shown in Fig. 4 and open-loop step-responses depicted in Fig. 5 it is evident that the behaviour of the process is similar to that of the second-order system whose time constant and stationary gain vary smoothly as functions of the state. Hence, the nonlinear polynomial model

$$\begin{aligned} y(k) &= (b_{10} + b_{11}y(k-1))u(k-1) + (b_{20} + b_{21}y(k-1))u(k-2) \\ &\quad -(a_{10} + a_{11}y(k-1))y(k-1) - (a_{20} + a_{21}y(k-1))y(k-2) \end{aligned} \tag{55}$$

is found. The model is similar to the linear one (54) but its parameters are linear functions of the output. Fig. 13 shows the output of the process and the output of the model for both training and test data sets. For the training data set SSE = 7.742230, for the test data set SSE = 6.425223. On the one hand, the accuracy of the nonlinear polynomial model is much higher in comparison with the linear model. On the other hand, the nonlinear polynomial model is significantly less precise than the neural one.

Considering Figs. 11–13 one can easily see that the neural model outperforms both linear and nonlinear polynomial models. The neural model is able to very precisely predict behaviour of the process. Hence, the pruned neural model with $K = 5$ hidden nodes is recommended to be next used in nonlinear MPC algorithms.

3.3. Yeast fermentation reactor control

The fundamental model (38)–(49) is used as the real process during simulations of MPC algorithms. The model is solved using the specialised solver ODE23S for stiff differential equations [33]. The horizons of MPC are $N = 10$, $Nu = 3$, the weighting coefficients $\lambda_p = 2$. (As far as choosing parameters of MPC, there are many tuning criteria in the literature [4,34] and this issue is not discussed here.) The manipulated variable is constrained, $F_{ag}^{min} = 0$ l/h, $F_{ag}^{max} = 200$ l/h. In all considered MPC algorithms two reference trajectories are used. The first trajectory is

$$T_r^{ref}(k) = \begin{cases} T_{r,0} & \text{if } k < 3 \\ 28.75\,^{\circ}\text{C} & \text{if } 3 \le k \le 49 \\ 28.0\,^{\circ}\text{C} & \text{if } 50 \le k \le 99 \\ 28.75\,^{\circ}\text{C} & \text{if } 100 \le k \le 149 \\ T_{r,0} & \text{if } 150 \le k \le 200 \end{cases} \tag{56}$$

while the second trajectory is

$$T_r^{ref}(k) = \begin{cases} T_{r,0} & \text{if } k < 3 \\ 30.25\,^{\circ}\text{C} & \text{if } 3 \le k \le 49 \\ 31.0\,^{\circ}\text{C} & \text{if } 50 \le k \le 99 \\ 30.25\,^{\circ}\text{C} & \text{if } 100 \le k \le 149 \\ T_{r,0} & \text{if } 150 \le k \le 200 \end{cases} \tag{57}$$

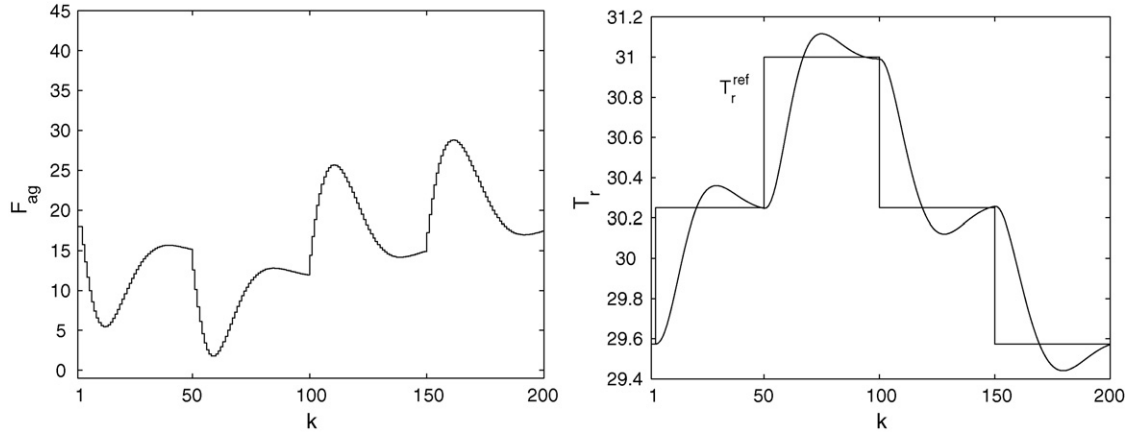

Fig. 15. Experiment 1 (the reference trajectory 2): the GPC algorithm based on the linear model.

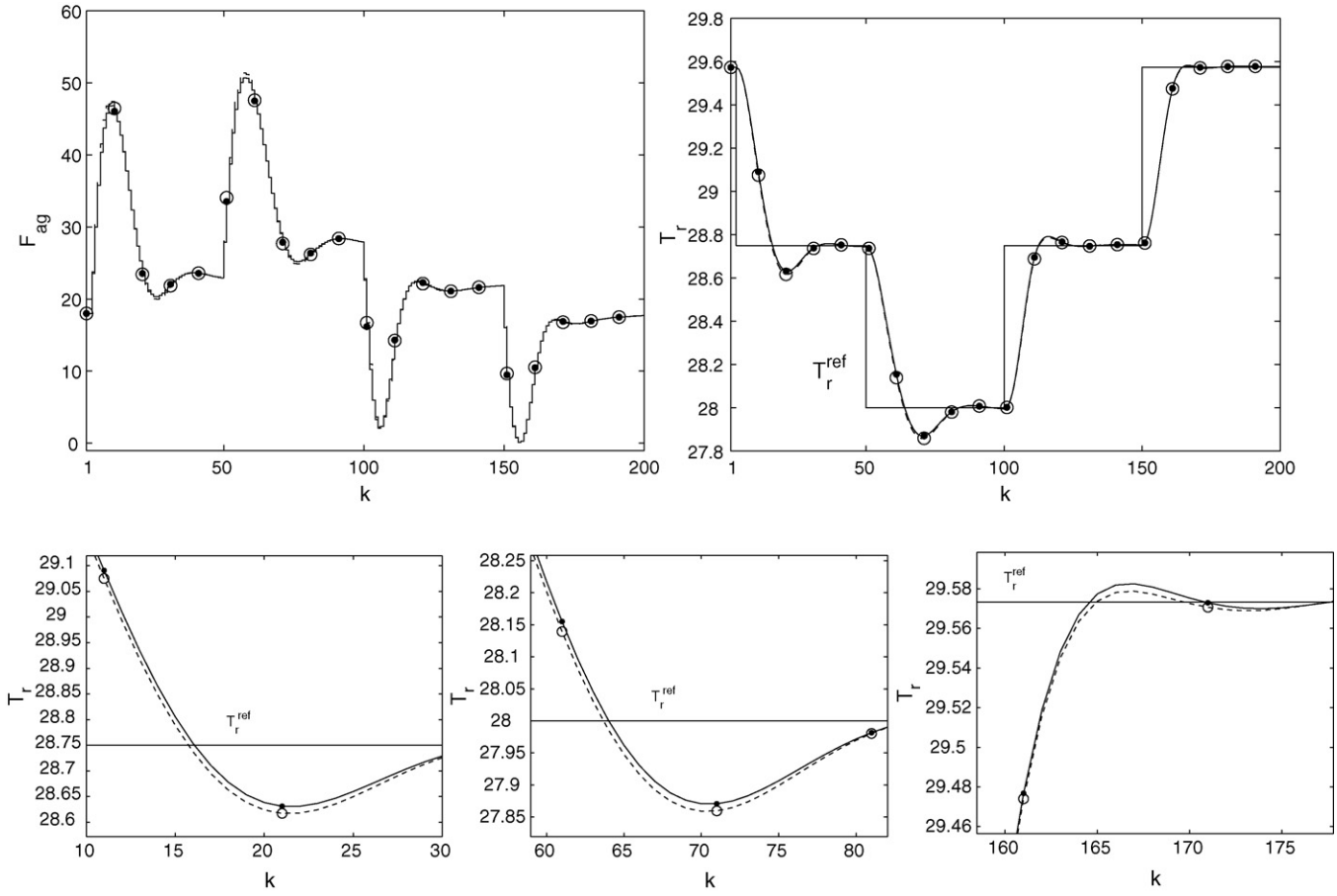

Fig. 16. Experiment 1 (the reference trajectory 1): the MPC-NPL algorithm (*dashed line with circles*) and the MPC-NO algorithm (*solid line with dots*) based on the same neural model; *top*: the whole simulation, *bottom*: enlarged fragments.

To emphasise the accuracy and the computational efficiency of the discussed MPC-NPL algorithm based on the neural model, in the following part of the article three MPC algorithms are compared

(a) the MPC algorithm based on the linear model (54);
(b) the suboptimal MPC-NPL algorithm based on the pruned neural model (51);
(c) the MPC algorithm with on-line Nonlinear Optimisation (MPC-NO) [17,36,37] based on the same neural model.

As the first MPC algorithm the Generalized Predictive Control (GPC) algorithm [3] is used. At each sampling instant of the algorithm a quadratic programming problem is solved. The MPC-NPL algorithm uses the quadratic programming procedure whereas the MPC-NO algorithm uses Sequential Quadratic Programming (SQP) [1] nonlinear optimization subroutine. As the initial point for optimisation, $N_u - 1$ control values calculated at the previous sampling instant and not applied to the process is used.

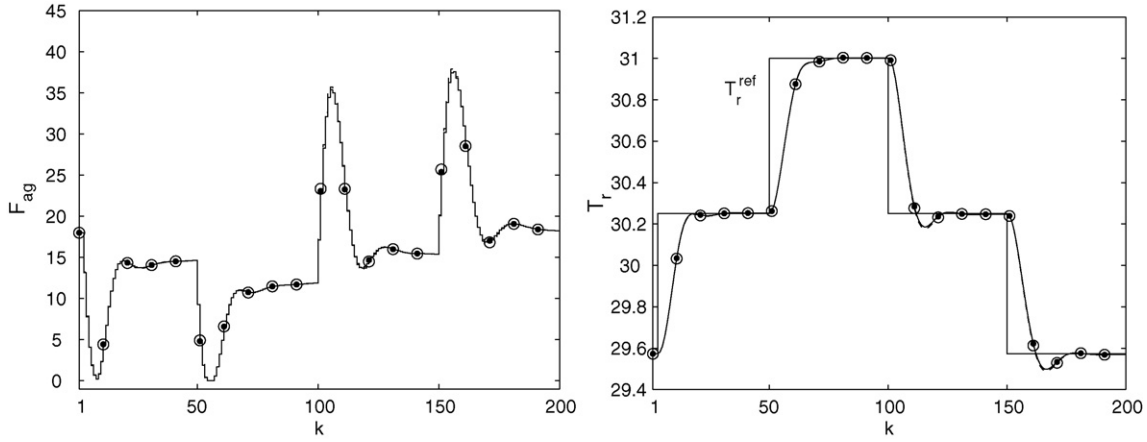

Fig. 17. Experiment 1 (the reference trajectory 2): the MPC-NPL algorithm (*dashed line with circles*) and the MPC-NO algorithm (*solid line with dots*) based on the same neural model.

Table 4
The performance index J_1 for GPC, MPC-NPL and MPC-NO algorithms

Experiment	GPC	MPC-NPL	MPC-NO
1	85.742	49.586	49.363
2	85.742	58.096	57.883
3	86.090	51.885	51.743
4	78.871	51.932	51.769
5	83.499	53.265	53.024
6	86.221	55.564	55.385

Table 5
The performance index J_2 for GPC, MPC-NPL and MPC-NO algorithms

Experiment	GPC	MPC-NPL	MPC-NO
1	40.417	25.950	25.893
2	40.417	31.911	31.892
3	40.486	26.070	25.991
4	35.446	26.693	26.620
5	37.757	26.589	26.586
6	41.360	26.954	26.900

3.3.1. Experiment 1

In the first experiment carried out it is assumed that unmeasured disturbances are not present and the temperature T_{in} of the substrate flow entering the reactor (the main disturbance of the process) has its nominal value 25 °C.

Because of a highly nonlinear nature of the yeast fermentation reactor, the accuracy of the linear model is low as shown in Fig. 12. As a results, the GPC algorithm based on the linear model is unable to control the process efficiently as depicted in Figs. 14 and 15. The linear GPC algorithm is very slow, the manipulated variable is quite sluggish, rendering the controlled variable unable to track reference trajectories fast enough. Hence, it is justified to use nonlinear models in MPC rather than linear ones.

Simulation results of the MPC-NPL algorithm and the MPC-NO algorithm based on the same neural model are depicted in Figs. 16 and 17. Both MPC algorithms based on the neural model are significantly faster than the GPC algorithm based on the linear model. Moreover, for both reference trajectories the closed-loop performance obtained in the suboptimal MPC-NPL algorithm with quadratic programming is very similar to that obtained in the computationally demanding MPC-NO approach, in which a nonlinear optimisation problem has to be solved on-line at each sampling instant.

Differences between MPC-NPL and MPC-NO approaches are very small, which means that obtained trajectories of the system are practically the same as shown in 16 and Fig. 17. In order to make it possible to compare all three examined algorithms (GPC, MPC-NPL and MPC-NO), two typical control performance indices are calculated after completing simulations. The sum of absolute values of differences between the reference trajectory and the actual value of the controlled variable over the whole simulation horizon

$$J_1 = \sum_{k=1}^{k=200} |T_r^{ref}(k) - T_r(k)| \tag{58}$$

and the sum of squared differences

$$J_2 = \sum_{k=1}^{k=200} (T_r^{ref}(k) - T_r(k))^2 \tag{59}$$

are calculated. Parameters J_1 and J_2 obtained in all six considered experiments are collected in Tables 4 and 5, performance indices are calculated for both reference trajectories.

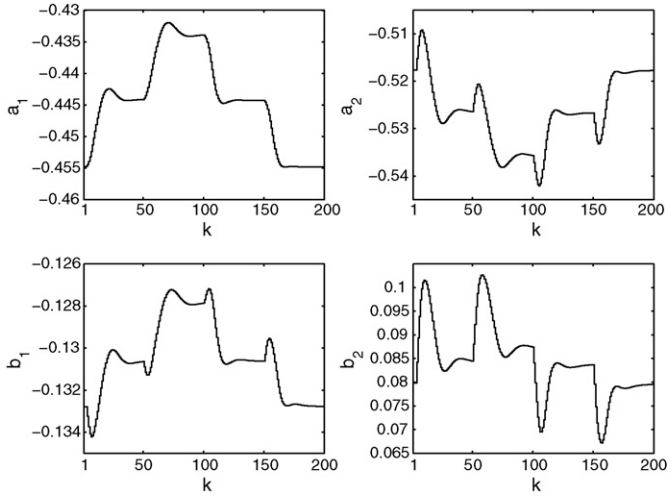

Fig. 18. Experiment 1 (the reference trajectory 1): parameters of the linearised model in the MPC-NPL algorithm.

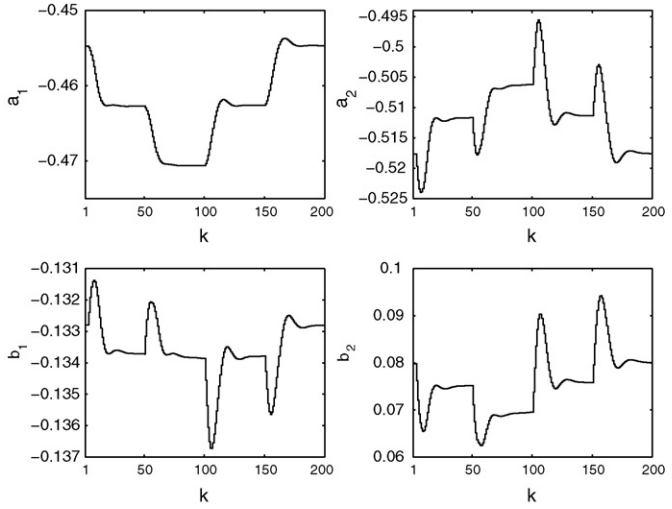

Fig. 19. Experiment 1 (the reference trajectory 2): parameters of the linearised model in the MPC-NPL algorithm.

Table 6
The computational complexity in terms of floating point operations (MFLOPS) of MPC-NO and MPC-NPL algorithms

Algorithm	N	$N_u = 1$	$N_u = 2$	$N_u = 3$	$N_u = 4$	$N_u = 5$	$N_u = 10$	$N_u = 15$
MPC-NPL	5	0.57	0.68	0.93	1.29	1.81	–	–
MPC-NO	5	4.10	6.19	13.25	17.71	28.65	–	–
MPC-NPL	10	0.90	1.03	1.31	1.70	2.26	7.88	–
MPC-NO	10	6.28	10.87	19.10	28.50	43.52	180.53	–
MPC-NPL	15	1.23	1.38	1.68	2.11	2.70	8.60	21.83
MPC-NO	15	8.92	17.93	28.61	42.20	61.64	222.34	587.43

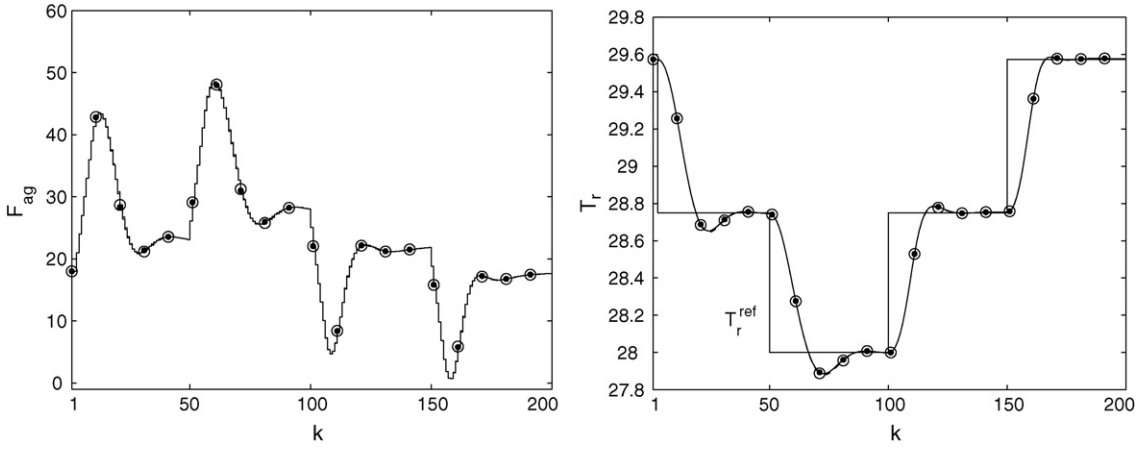

Fig. 20. Experiment 2 (the reference trajectory 1): the MPC-NPL algorithm (*dashed line with circles*) and the MPC-NO algorithm (*solid line with dots*) based on the same neural model with constraints imposed on increments of the manipulated variable, $\Delta F_{ag}^{max} = 3$ l/h.

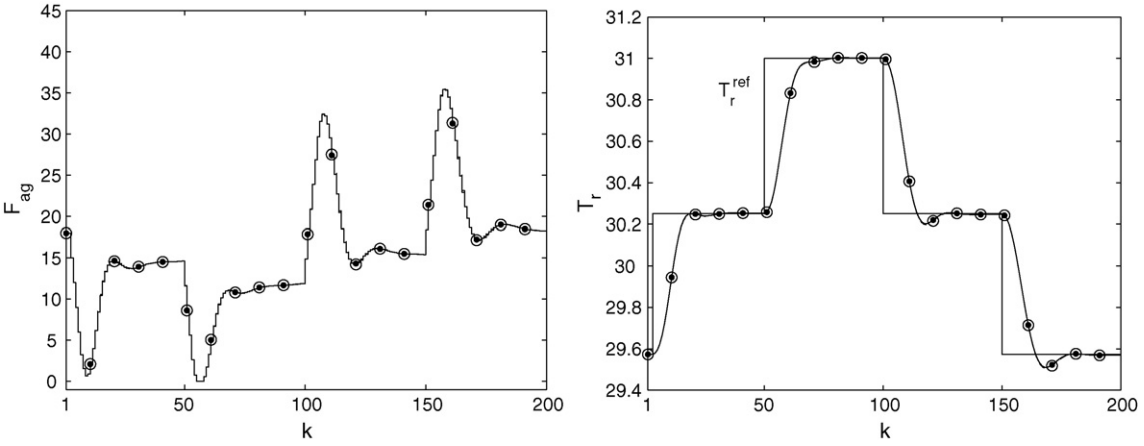

Fig. 21. Experiment 2 (the reference trajectory 2): the MPC-NPL algorithm (*dashed line with circles*) and the MPC-NO algorithm (*solid line with dots*) based on the same neural model with constraints imposed on increments of the manipulated variable, $\Delta F_{ag}^{max} = 3$ l/h.

Table 6 shows the computational complexity of compared MPC-NPL and MPC-NO algorithms based on the neural model in terms of floating point operations (MFLOPS) for different combinations of prediction and control horizons (for both reference trajectories). In general, the suboptimal MPC-NPL algorithm is many times computationally less demanding that the MPC-NO algorithm. The control horizon has significantly bigger impact on the computational burden than the prediction horizon since N_u is the number of decision variables of the optimisation problem.

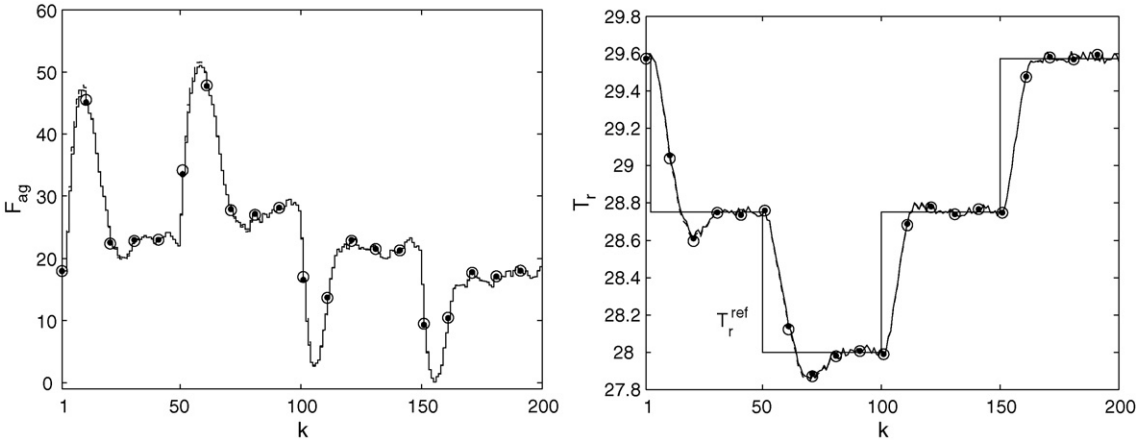

Fig. 22. Experiment 3 (the reference trajectory 1): the MPC-NPL algorithm (*dashed line with circles*) and the MPC-NO algorithm (*solid line with dots*) based on the same neural model in presence of unmeasured disturbances.

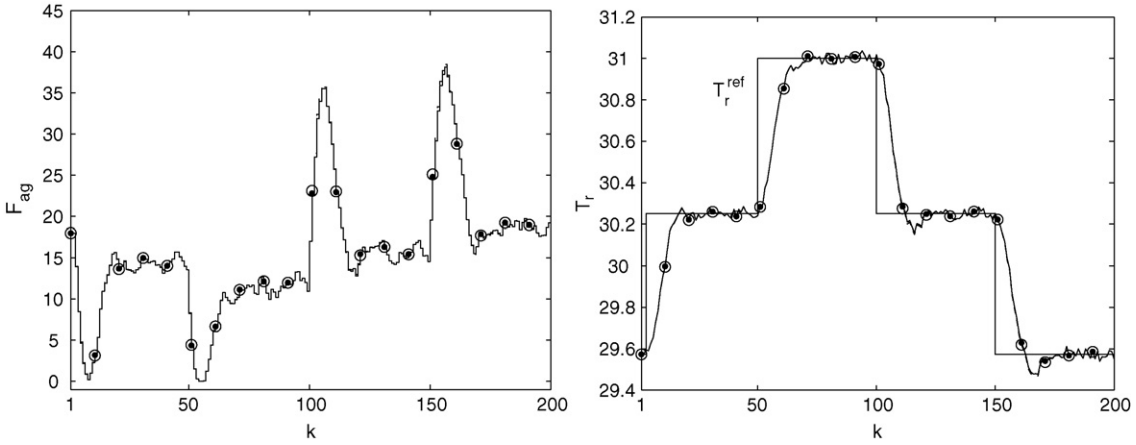

Fig. 23. Experiment 3 (the reference trajectory 2): the MPC-NPL algorithm (*dashed line with circles*) and the MPC-NO algorithm (*solid line with dots*) based on the same neural model in presence of unmeasured disturbances.

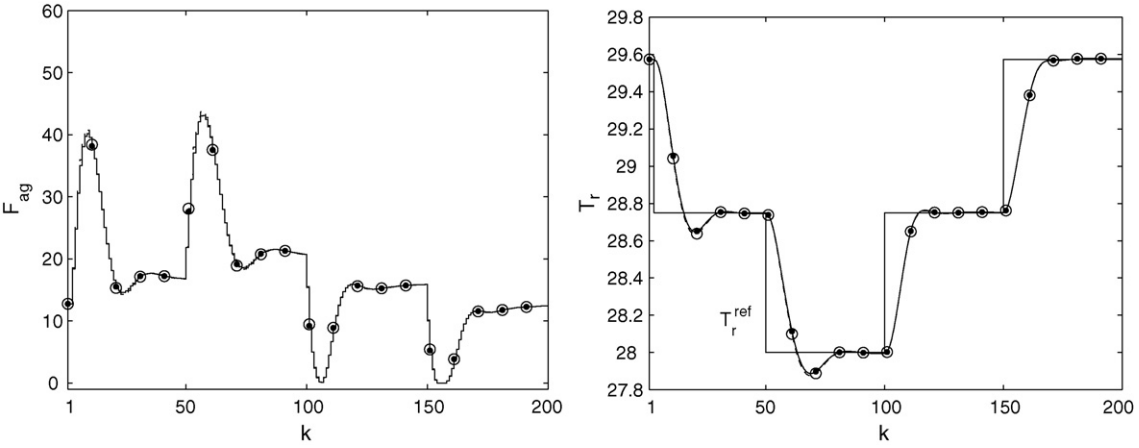

Fig. 24. Experiment 4 (the reference trajectory 1): the MPC-NPL algorithm (*dashed line with circles*) and the MPC-NO algorithm (*solid line with dots*) based on the same neural model, the temperature T_{in} of the substrate flow entering the reactor is 24 °C.

In the MPC-NPL algorithm the neural model (7) is linearised on-line about the current state of the process (31). The linearised model (32) is next used for calculation of the current value of the manipulated variable as described in Section 2.3. Parameters a_1, a_2, b_1, b_2 of the linearised model are shown in Figs. 18 and 19. One can observe that changes of these parameters are not big. It is because reference trajectories (56) and (57) are defined in such a way that $28\,^{\circ}\mathrm{C} \leq T_r^{ref} \leq 31\,^{\circ}\mathrm{C}$ whereas the steady-state characteristic shown in Fig. 4 covers the whole range of the manipulated variable ($0\,\mathrm{l/h} \leq F_{ag} \leq 200\,\mathrm{l/h}$) for which $22.52\,^{\circ}\mathrm{C} \leq T_r \leq 36.03\,^{\circ}\mathrm{C}$. Naturally, the neural model covers the whole range of interest.

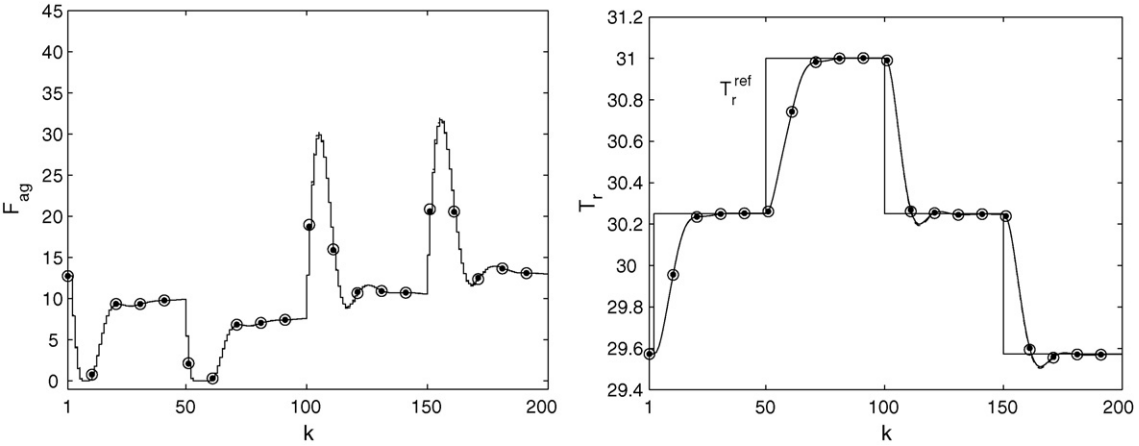

Fig. 25. Experiment 4 (the reference trajectory 2): the MPC-NPL algorithm (*dashed line with circles*) and the MPC-NO algorithm (*solid line with dots*) based on the same neural model, the temperature T_{in} of the substrate flow entering the reactor is 24 °C.

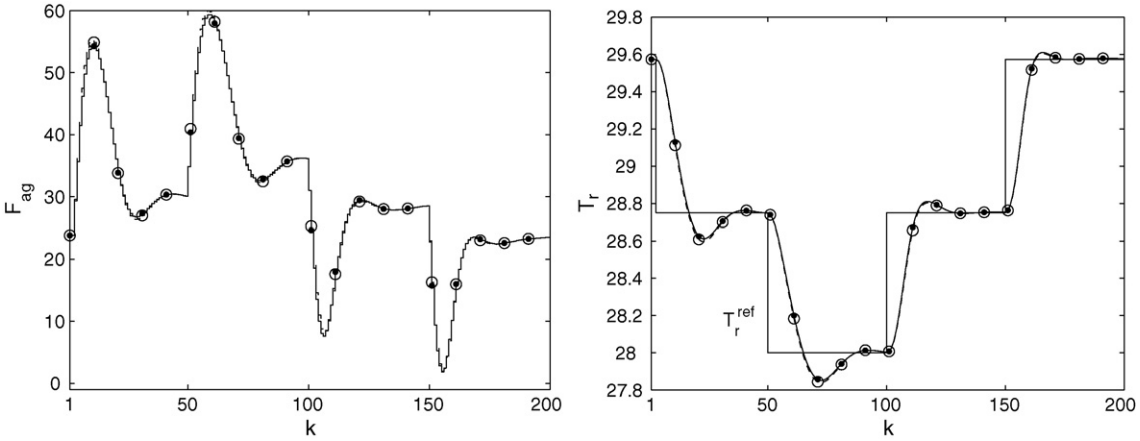

Fig. 26. Experiment 5 (the reference trajectory 1): the MPC-NPL algorithm (*dashed line with circles*) and the MPC-NO algorithm (*solid line with dots*) based on the same neural model, the temperature T_{in} of the substrate flow entering the reactor is 26 °C.

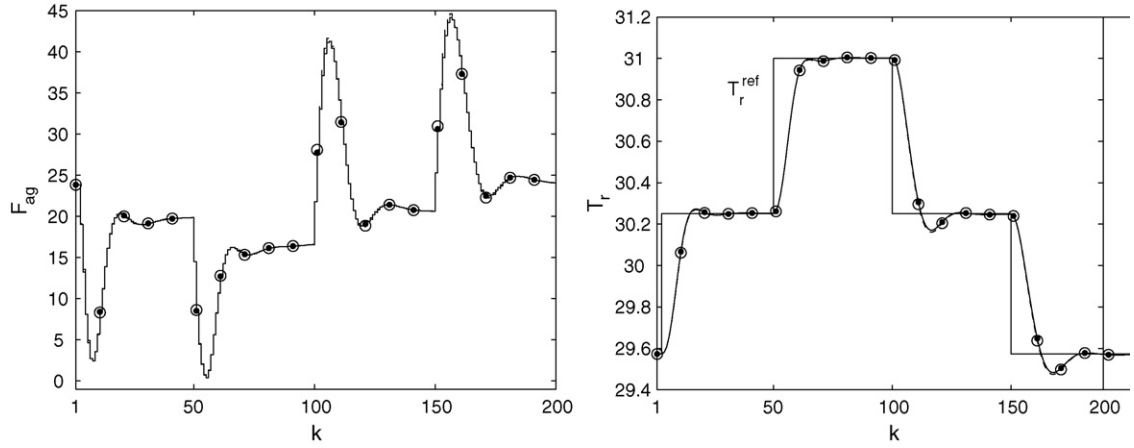

Fig. 27. Experiment 5 (the reference trajectory 2): the MPC-NPL algorithm (*dashed line with circles*) and the MPC-NO algorithm (*solid line with dots*) based on the same neural model, the temperature T_{in} of the substrate flow entering the reactor is 26 °C.

3.3.2. Experiment 2

In the second experiment, it is assumed that unmeasured disturbances are not present and the temperature T_{in} has its nominal value, but constraints imposed on increments of the manipulated variable are taken into account ($\Delta F_{ag}^{max} = 3$ l/h is assumed). Such constraints are likely to be very important when changes in the reference trajectory are big, they take into account the actuator's limitations. The GPC algorithm based on the linear model gives very slow transient responses, exactly the same as in the first experiment (Figs. 14 and). It is because increments of the manipulated variable are smaller than 3 l/h. Simulation results of both nonlinear MPC algorithms are shown in Figs. 20 and 21. In comparison with

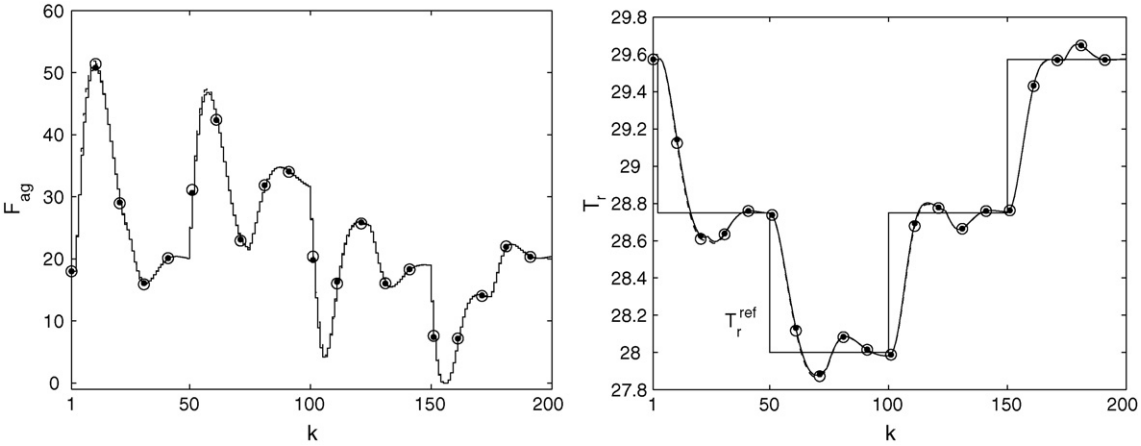

Fig. 28. Experiment 6 (the reference trajectory 1): the MPC-NPL algorithm (*dashed line with circles*) and the MPC-NO algorithm (*solid line with dots*) based on the same neural model in presence of step changes of the temperature T_{in} of the substrate flow entering the reactor given by (60).

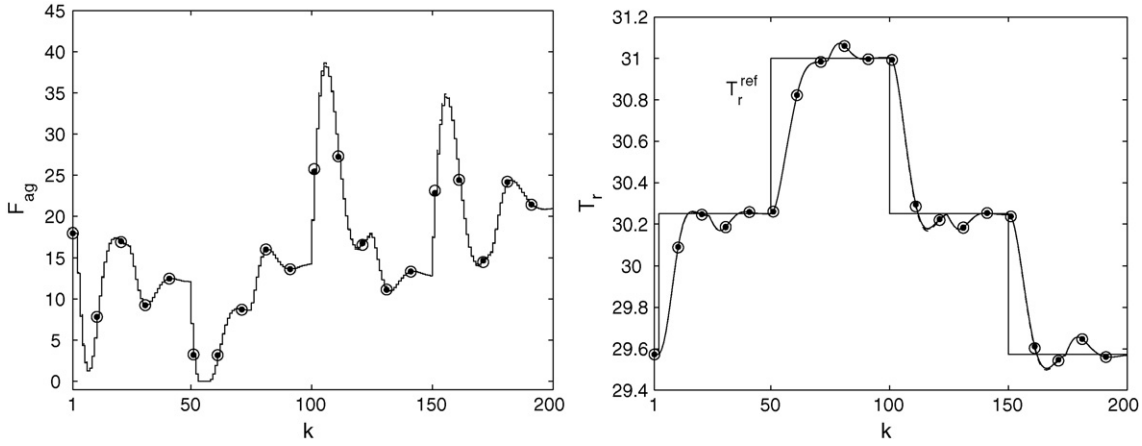

Fig. 29. Experiment 6 (the reference trajectory 2): the MPC-NPL algorithm (*dashed line with circles*) and the MPC-NO algorithm (*solid line with dots*) based on the same neural model in presence of step changes of the temperature T_{in} of the substrate flow entering the reactor given by (60).

Figs. 16 and 17, the additional constraints result in a slightly slower output profile, but the constraints are always satisfied. Constraints satisfaction is a great advantage when one compares the MPC algorithm described here and an easy to implement controller based on an inverse model of the process [26].

3.3.3. Experiment 3

In the third experiment, it is assumed that unmeasured disturbances affect the process and the temperature T_{in} has its nominal value. Disturbance rejection is a very important ability of any control algorithms as disturbances are unavoidable in practice. Simulation results are shown in Figs. 22 and 23. Because the GPC algorithm based on the linear model is slow, for this experiment and the the next ones simulation results of nonlinear MPC-NPL and MPC-NO algorithms are only depicted while performance indices J_1 and J_2 given in Tables 4 and 5 are calculated for all three studied algorithms.

3.3.4. Experiment 4

In the fourth experiment it is assumed that unmeasured disturbances are not present and the temperature T_{in} of the substrate flow entering the reactor (the main disturbance of the process) is decreased from its nominal value 25 ° C to 24 °C. In MPC algorithms still the same neural model is used (trained for nominal conditions). Simulation results are shown in Figs. 24 and 25. In comparison with the nominal value of the temperature T_{in} case (Figs. 16 and 17) the main change is the level of the manipulated variable.

3.3.5. Experiment 5

In the fifth experiment it is assumed that the temperature T_{in} of the substrate flow entering the reactor is increased from its nominal value 25 ° C to 26 °C. Figs. 26 and 27 depict simulation results.

3.3.6. Experiment 6

In the sixth experiment it is assumed that the temperature T_{in} of the substrate flow entering the reactor changes changes according to the equation

$$T_{in}(k) = \begin{cases} 25.0\,^{\circ}\mathrm{C} & \text{if } k < 3 \\ 25.5\,^{\circ}\mathrm{C} & \text{if } 3 \le k \le 24 \\ 24.5\,^{\circ}\mathrm{C} & \text{if } 25 \le k \le 74 \\ 25.5\,^{\circ}\mathrm{C} & \text{if } 75 \le k \le 124 \\ 24.5\,^{\circ}\mathrm{C} & \text{if } 125 \le k \le 174 \\ 25.5\,^{\circ}\mathrm{C} & \text{if } 175 \le k \le 200 \end{cases} \quad (60)$$

Figs. 28 and 29 depict simulation results.

Considering simulation results (figures and values of the performance indices J_1, J_2), one can conclude that the GPC algorithm based on the linear model is slow and the suboptimal MPC-NPL algorithm with quadratic programming gives very similar transient responses to those obtained in the computationally demanding MPC-NO approach, in which a nonlinear optimisation problem has to be solved on-line at each sampling instant.

4. Conclusions

This paper discusses the application of artificial neural networks to modelling and temperature control of a yeast fermentation biochemical reactor detailed in [26]. Because the process exhibit significantly nonlinear behaviour, the accuracy of the linear model is low. As a result, the classical PID controller and the MPC algorithm based on a linear model are unable to control the process efficiently as demonstrated in [26] and in this paper.

The identification procedure of neural models described in this paper contains two stages. At first, a few neural networks (with different numbers of hidden nodes) are trained using available data sets generated from the fundamental model. The model containing 5 hidden nodes is chosen as a reasonable compromise between the accuracy and the complexity. The rudimentary model has 31 weights. Next, in order to reduce the complexity of the neural model and to improve its prediction ability the neural network is pruned using the Optimal Brain Damage algorithm. As a result of pruning, 12 weights are removed, which means that the complexity of the rudimentary neural network is reduced by 38%. At the same time, the pruned model has very good generalisation abilities, i.e. the SSE performance index of the pruned neural network for the training data set is significantly smaller than that of the rudimentary network.

Next, a computationally efficient nonlinear MPC algorithm with Nonlinear Prediction and Linearisation (MPC-NPL) [17,18,36,37] is applied to the process. It is demonstrated that the algorithm results in closed-loop control performance similar to that obtained in nonlinear MPC, in which nonlinear optimisation is repeated at each sampling instant. The computational efficiency of the MPC-NPL algorithm is twofold. First of all, the suboptimal algorithm uses on-line only the numerically reliable quadratic programming procedure, unlike the nonlinear optimisation, which may terminate in a local minimum, the global solution is always found within a foreseeable time frame. Secondly, the computational complexity reduction obtained in the suboptimal algorithm compared to the MPC-NO algorithm in very big as shown in Table 6.

In this paper MPC algorithms based on neural models are recommended. Such an approach has many advantages. First of all, neural models are trained from given sets of process data, it is not necessary to develop complicated fundamental models (in this study the fundamental model is only used as the real process during simulations). Secondly, implementation of the considered MPC-NPL algorithm is relatively easy. It is because neural networks have simple structures and limited numbers of parameters. Moreover, neural models directly describe input–output relations of process variables. It means that complicated systems of differential and algebraic equations comprising the fundamental model do not have to be solved on-line in MPC. It is particulary important in the case of the considered yeast fermentation process because its fundamental model is stiff, the specialised solver for stiff differential equations should be used.

References

[1] M.S. Bazaraa, J. Sherali, K. Shetty, Nonlinear Programming: Theory and Algorithms, Prentice Hall, 1993.
[2] B.M. Åkesson, H.T. Toivonen, A neural network model predictive controller, J. Proc. Control 16 (3) (2006) 937–946.
[3] D.W. Clarke, C. Mohtadi, P.S. Tuffs, Generalized predictive control—I. The basic algorithm, II. Extensions and interpretations, Automatica 23 (2) (1987) 137–160.
[4] D.W. Clarke, C. Mohtadi, Properties of Generalized Predictive Control, Automatica 25 (6) (1989) 859–875.
[5] Haber, R., Unbehauen, H., Structure identification of nonlinear dynamic systems - a survey of input/output approaches, Automatica 26, 651–677.
[6] L.K. Hansen, M.W. Pedersen, Controlled growth of cascade correlationnets in: M. Marino, P.G. Morasso (Eds.), Proceedings of the International Conference on Artificial Neural Networks, ICANN 1994, Sorento, Italy, pp. 797–800.
[7] B. Hassibi, B. Stork, Second order derivatives for network prunning: Optimal brain surgeon, in: D. Touretzky (Ed.), Advances of NIPS5, Morgan Kaufmann, San Mateo, 1993, pp. 164–171.
[8] S. Haykin, Neural Networks– a Comprehensive Foundation, Prentice Hall, Englewood Cliffs, 1999.
[9] M.A. Henson, Nonlinear model predictive control: current status and future directions, Comput. Chem. Eng. 23 (2) (1998) 187–202.
[10] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366.
[11] Huang, Y., Lu, W. M., Nonlinear optimal control: alternatives to Hamilton-Jacobi equation, Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan, 1996. 3942–3947.
[12] M.A. Hussain, Review of the applications of neural networks in chemical process control– simulation and online implmementation, Artif. Intell. Eng. 13 (1) (1999) 55–68.
[13] Y. LeCun, J. Denker, S. Solla, D. Touretzky, Optimal Brain Damage. Advances of NIPS2, Morgan Kaufmann, San Mateo, 1990, pp. 598–605.
[14] G.P. Liu, V. Kadirkamanathan, S.A. Billings, Predictive control for non-linear systems using neural networks, Int. J. Control 71 (6) (1998) 1119–1132.
[15] W.L. Luyben, Process Modelling, Simulation, Control for Chemical Engineers, McGraw Hill, 1990.
[16] M. Ławryńczuk, W. Tadej, A computationally efficient stable dual-mode type nonlinear predictive control algorithm, Control Cybernet. 37 (1) (2008) 99–132.
[17] M. Ławryńczuk, A family of model predictive control algorithms with artificial neural networks, Int. J. Appl. Math. Comp. Sci. 17 (2) (2007) 217–232.
[18] M. Ławryńczuk, P. Tatjewski, An efficient nonlinear predictive control algorithm with neural models and its application to a high-purity distillation process, Lecture Notes in Artificial Intelligence 4029 (2006) 76–85 (Rutkowski, L., Tadeusiewicz, R., Zadeh, L. A., Zurada, J. M. (eds.): Proceedings of the Eighth International Conference on Artificial Intelligence and Soft Computing, ICAISC 2006, Zakopane, Poland).
[19] J.M. Maciejowski, Predictive Control with Constraints, Prentice Hall, Harlow, 2002.
[20] D.Q. Mayne, J.B. Rawlings, C.V. Rao, P.O.M. Scokaert, Constrained model predictive control:stability and optimality, Automatica 36 (6) (2000) 789–814.
[21] T.E. Marlin, Process Control, McGraw-Hill, New York, 1995.
[22] P. Marusak, P. Tatjewski, Stability analysis of nonlinear control systems with unconstrained fuzzy predictive controllers, Arch. Control Sci. 12 (2002) 267–288.
[23] H. Michalska, D.Q. Mayne, Robust receding horizon control of constrained nonlinear systems, IEEE Trans. Automatic Control 38 (11) (1993) 1623–1633.
[24] M. Morari, J.H. Lee, Model predictive control: past, present and future, Comput. Chem. Eng. 23 (4/5) (1999) 667–682.
[25] M. Nørgaard, O. Ravn, N.K. Poulsen, L.K. Hansen, Neural Networks for Modelling and Control of Dynamic Systems, Springer, London, 2000.
[26] Z.K. Nagy, Model based control of a yeast fermentation bioreactors using optimally designed artificial neural networks, Chem. Eng. J. 127 (1) (2007) 95–109.
[27] R.H. Nyström, Åkesson, B. M., Toivonen, H. T., Gain-scheduling controllers based on velocity-form linear parameter-varying models applied to an example process, Ind. Eng. Chem. Res. 41 (2002) 220–229.
[28] T. Parisini, M. Sanguineti, R. Zoppoli, Nonlinear stabilization by receding-horizon neural regulators, Int. J. Control 70 (3) (1998) 341–362.
[29] S. Piche, B. Sayyar-Rodsari, D. Johnson, M. Gerules, Nonlinear model predictive control using neural networks, IEEE Control Syst. Mag. 20 (3) (2000) 56–62.
[30] M. Pottmann, D.E. Seborg, A nonlinear predictive control strategy based on radial basis function models, Comput. Chem. Eng. 21 (9) (1997) 965–980.
[31] S.J. Qin, T.A. Badgwell, A survey of industrial model predictive control technology, Control Eng. Pract. 11 (7) (2003) 733–764.
[32] J.A. Rossiter, Model-Based Predictive Control, CRC Press, Boca Raton, 2003.
[33] L.F. Shampine, M.W. Reichelt, The MATLAB ODE suite, SIAM J. Sci. Comp. 18 (1997) 1–22.
[34] R. Scattolini, S. Bittanti, On the choice of the horizon in long-range predictive control—some simple criteria, Automatica 26 (5) (1990) 915–917.
[35] K.S. Strömborg, H.T. Toivonen, K.E. Häggblom, K.V. Waller, Multivariable nonlinear and adaptive control of a distillation column, AIChE J. 41 (1995) 195–199.
[36] P. Tatjewski, Advanced Control of Industrial Processes, Structures and Algorithms, Springer, London, 2007.
[37] P. Tatjewski, Ławryńczuk, M. Soft computing in model-based predictive control, Int. J. Appl. Math. Comp. Sci. 16 (1) (2006) 101–120.
[38] Z. Trajanoski, P. Wach, Neural predictive control for insulin delivery using the subcutaneous route, IEEE Trans. Biomed. Eng. 45 (9) (1998) 1122–1134.
[39] D.L. Yu, J.B. Gomm, Implementation of neural network predictive control to a multivariable chemical reactor, Control Eng. Practice 11 (11) (2003) 1315–1323.