

# Nonlinear predictive control of dynamic systems represented by Wiener–Hammerstein models

Maciej Ławryńczuk

Received: 17 December 2015 / Accepted: 12 July 2016 / Published online: 15 August 2016  
© The Author(s) 2016. This article is published with open access at Springerlink.com

**Abstract** This paper is concerned with computationally efficient nonlinear model predictive control (MPC) of dynamic systems described by cascade Wiener–Hammerstein models. The Wiener–Hammerstein structure consists of a nonlinear steady-state block sandwiched by two linear dynamic ones. Two nonlinear MPC algorithms are discussed in details. In the first case the model is successively linearised on-line for the current operating conditions, whereas in the second case the predicted output trajectory of the system is linearised along the trajectory of the future control scenario. Linearisation makes it possible to obtain quadratic optimisation MPC problems. In order to illustrate efficiency of the discussed nonlinear MPC algorithms, a heat exchanger represented by the Wiener–Hammerstein model is considered in simulations. The process is nonlinear, and a classical MPC strategy with linear process description does not lead to good control result. The discussed MPC algorithms with on-line linearisation are compared in terms of control quality and computational efficiency with the fully fledged nonlinear MPC approach with on-line nonlinear optimisation.

**Keywords** Process control · Model predictive control · Wiener–Hammerstein systems · Optimisation · Linearisation

---

M. Ławryńczuk (✉)  
Institute of Control and Computation Engineering, Faculty  
of Electronics and Information Technology, Warsaw  
University of Technology, Warsaw, Poland  
e-mail: M.Lawrynczuk@ia.pw.edu.pl

## 1 Introduction

The dynamic model of the controlled process is used only during development of the classical controllers, e.g. linear quadratic regulator (LQR). A conceptually different approach is a model predictive control (MPC) algorithm, in which the model is used on-line in order to predict the future behaviour of the process and determine the optimal control policy [1]. Typically, the objective of MPC is to minimise the predicted deviations between the output (or state) trajectory and the set-point trajectory. The MPC algorithms have the unique ability to efficiently take into account constraints imposed on process inputs (manipulated variables) and outputs (controlled variables) or state variables. Furthermore, they may be applied not only to single-input single-output processes, but also to complex multiple-input multiple-output systems with strong cross-couplings. MPC algorithms make it possible to efficiently control nonminimum phase processes, e.g. with long time delays. As a result, the MPC algorithms are very frequently used in practice in different areas, primarily in chemical, petrochemical, food and paper industries [2]. Applications of MPC algorithms include: manipulators [3], active steering systems in cars [4], air conditioning systems [5], anti-lock brake systems in cars [6], chemical reactors [7], distillation columns [8], overhead cranes [9]. Examples of less typical applications of MPC are: active queue management in TCP/IP networks [10] and drinking water networks [11].

The classical formulations of MPC algorithms, e.g. dynamic matrix control (DMC) and generalised predictive control (GPC) [1], use for prediction linear models. Although they are successful in many applications, a large class of industrial processes are nonlinear and in such cases the classical MPC algorithms usually cannot give acceptable control quality. In nonlinear MPC the basic question is the choice of the model structure, which is next used on-line for prediction and optimisation of the future control policy. Although many implementations of MPC use black-box models, e.g. fuzzy structures [5] or neural networks [12], a viable alternative is to use block-oriented cascade models which are composed of linear dynamic parts and nonlinear steady-state parts [13, 14]. It is necessary to point out that the cascade model representation is a straightforward choice in case of many dynamic processes.

The simplest two-block cascade models are very frequently used for control, especially for MPC. MPC algorithms for the Hammerstein structure (a nonlinear steady-state block followed by a linear dynamic one) discussed in [15–17] use an inverse of the steady-state part of the model to compensate for process nonlinearity. It is also possible to find on-line a linear approximation of the model for current operating conditions and next use the linearised model in MPC [12, 18] or find directly a linear approximation of the predicted trajectory [12]. MPC algorithms for the Wiener structure (a linear dynamic block followed by a nonlinear steady-state one) with an inverse of the steady-state part are discussed in [19–21], MPC approaches with on-line model linearisation are discussed in [12, 22, 23], and MPC approaches with on-line trajectory linearisation are discussed in [12, 23]. In case of the three-block cascade models of the Hammerstein–Wiener type (a linear dynamic block sandwiched by two nonlinear steady-state ones), MPC algorithms with an inverse of the steady-state parts are detailed in [24, 25], and MPC with on-line trajectory linearisation is discussed in [26]. In all the cited MPC algorithms application of an inverse model or on-line linearisation makes it possible to obtain a computationally simple MPC quadratic optimisation problems. MPC with transformation of the nonlinearities into polytopic descriptions which results in a convex optimisation task subject to linear matrix inequalities is considered in [27].

The cascade Wiener–Hammerstein structure consists of a nonlinear steady-state block sandwiched by two linear dynamic ones. It may be used to describe dif-

ferent dynamic processes, e.g. a superheater–desuperheater [28], an RF amplifier [29], a paralysed muscle under electrical stimulation [13], a heat exchanger [30], a DC–DC converter [31], an equaliser for optical communication [32], an electronic circuit [33]. Although control of dynamic systems represented by Wiener–Hammerstein models is discussed in the literature, e.g. extremum seeking control is described in [34], MPC of such systems has not been considered so far. This paper details two computationally efficient nonlinear MPC algorithms based on Wiener–Hammerstein models. In the first case the model is successively linearised on-line for the current operating conditions, whereas in the second case the predicted output trajectory of the system is linearised along the trajectory of the future control scenario. In both MPC approaches on-line linearisation makes it possible to obtain quadratic optimisation problems, which may be efficiently solved using the available solvers [35]. In order to illustrate efficiency of the discussed nonlinear MPC algorithms, a heat exchanger represented by the Wiener–Hammerstein model is considered in simulations. The process is nonlinear, and a classical MPC strategy in which a linear process description is used does not lead to good control. The discussed MPC algorithms with on-line linearisation are compared in terms of control quality and computational efficiency with the fully fledged nonlinear MPC approach with on-line nonlinear optimisation. This work extends the algorithms developed for Hammerstein, Wiener and Hammerstein–Wiener systems discussed in [12, 23, 26].

This paper is structured as follows. Section 2 reminds the idea of MPC, and Sect. 3 defines the structure of the Wiener–Hammerstein–Wiener model. Section 4 details the nonlinear MPC algorithms for Wiener–Hammerstein systems. Section 5 presents simulation results and their comparisons for a benchmark heat exchanger system. Finally, Sect. 6 concludes the paper.

## 2 Model predictive control problem formulation

Let  $u$  denote the input (manipulated) variable of the process and  $y$  denote the output (controlled) variable. In contrast to the classical control schemes, e.g. PID, in MPC algorithms [1] at each consecutive sampling instant  $k$  not only the current value of the manipulated

variable,  $u(k)$ , but a control policy defined for a control horizon  $N_u$  is calculated. Typically, control increments  $\Delta u(k) = [\Delta u(k|k) \Delta u(k+1|k) \dots \Delta u(k+N_u-1|k)]^T$  (1)

rather than values of the manipulated variable are determined, and the increments are defined as  $\Delta u(k|k) = u(k|k) - u(k-1)$ ,  $\Delta u(k+p|k) = u(k+p|k) - u(k+p-1|k)$  for  $p = 1, \dots, N_u - 1$ . It is assumed that  $\Delta u(k+p|k) = 0$  for  $p \geq N_u$ , i.e.  $u(k+p|k) = u(k+N_u-1|k)$  for  $p \geq N_u$ . The decision variables (1) of the MPC algorithm are calculated at each sampling instant from an optimisation problem in which the predicted control errors are minimised. They are defined as the differences between the set-point trajectory  $y^{sp}(k+p|k)$  and the predicted values of the process output, i.e.  $\hat{y}(k+p|k)$ , over the prediction horizon  $N \geq N_u$ , i.e. for  $p = 1, \dots, N$ . A dynamic model of the controlled process is used for prediction on-line. The MPC cost function is usually

$$J(k) = \sum_{p=1}^N (y^{sp}(k+p|k) - \hat{y}(k+p|k))^2 + \lambda \sum_{p=0}^{N_u-1} (\Delta u(k+p|k))^2$$

where  $\lambda > 0$  is a weighting coefficient which makes it possible not only to influence the speed of the algorithm, but also to assure good computational properties for an optimisation solver (tuning of MPC is discussed elsewhere [1]). Additionally, the optimisation algorithm may take into account some constraints imposed on the manipulated variable, which result from the physical limits of actuators, and on the predicted values of the controlled variable, which are usually enforced by some technological reasons. Typically, the MPC optimisation problem has the following form

$$\begin{aligned} & \min_{\Delta u(k|k), \dots, \Delta u(k+N_u-1|k)} \{J(k)\} \\ & \text{subject to} \\ & u^{\min} \leq u(k+p|k) \leq u^{\max}, \quad p = 0, \dots, N_u - 1 \\ & -\Delta u^{\max} \leq \Delta u(k+p|k) \leq \Delta u^{\max}, \\ & \quad p = 0, \dots, N_u - 1 \\ & y^{\min} \leq \hat{y}(k+p|k) \leq y^{\max}, \quad p = 1, \dots, N \end{aligned} \quad (2)$$

where  $u^{\min}$ ,  $u^{\max}$ ,  $\Delta u^{\max}$ ,  $y^{\min}$ ,  $y^{\max}$  define constraints imposed on: the magnitude of the manipulated (input)

variable, the increment of the input variable and the magnitude of the controlled (output) variable, respectively. If the output constraints are necessary, the MPC optimisation task (2) may be infeasible. A simple, but efficient method to solve the problem is to relax the output constraints by the so-called soft constraints, which may be temporarily relaxed [1]. Although in MPC at each sampling instant the whole sequence of control increments (1), of length  $N_u$ , is calculated, only the first element of that sequence is actually applied to the process, i.e.  $u(k) = \Delta u(k|k) + u(k-1)$ . At the next sampling instant,  $k+1$ , the prediction is shifted one step forward and the whole procedure is repeated.

### 3 Hammerstein–Wiener model of the process

The structure of the Wiener–Hammerstein model is depicted in Fig. 1. It consists of three separate blocks: two linear dynamic ones and a nonlinear steady-state one, which are connected in series in such a way that the nonlinear block is between the linear ones. The auxiliary signal between the input linear dynamic part and the nonlinear steady-state one is denoted by  $v$ , and the auxiliary signal between the nonlinear steady-state part and the output linear dynamic one is denoted by  $x$ . The first (input) linear dynamic part of the model is described by

$$A^1(q^{-1})v(k) = B^1(q^{-1})u(k) \quad (3)$$

where the polynomials in the backward shift operator ( $q^{-1}$ ) are

$$A^1(q^{-1}) = 1 + a_1^1 q^{-1} + \dots + a_{n_{A^1}}^1 q^{-n_{A^1}} \quad (4)$$

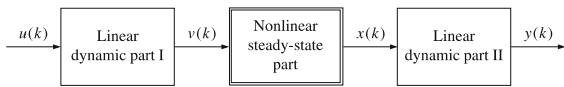
$$B^1(q^{-1}) = b_1^1 q^{-1} + \dots + b_{n_{B^1}}^1 q^{-n_{B^1}} \quad (5)$$

The order of dynamics of the first block is defined by the integers  $n_{A^1}$  and  $n_{B^1}$ . The nonlinear steady-state block is characterised by

$$x(k) = f(v(k)) \quad (6)$$

where it is assumed that the function  $f: \mathbb{R} \rightarrow \mathbb{R}$  is differentiable. The second (output) linear dynamic part of the model is defined by

$$A^2(q^{-1})y(k) = B^2(q^{-1})x(k) \quad (7)$$



**Fig. 1** The structure of the Wiener–Hammerstein model

where the polynomials are

$$A^2(q^{-1}) = 1 + a_1^2 q^{-1} + \dots + a_{n_{A^2}}^2 q^{-n_{A^2}} \tag{8}$$

$$B^2(q^{-1}) = b_1^2 q^{-1} + \dots + b_{n_{B^2}}^2 q^{-n_{B^2}} \tag{9}$$

and the order of dynamics of the second dynamic block is defined by the integers  $n_{A^2}$  and  $n_{B^2}$ .

From Eqs. (7)–(9), the output of the model is

$$y(k) = \sum_{i=1}^{n_{B^2}} b_i^2 x(k-i) - \sum_{i=1}^{n_{A^2}} a_i^2 y(k-i) \tag{10}$$

where from Eq. (6) the auxiliary signal  $x$  at the previous sampling instants is

$$x(k-i) = f(v(k-i)) \tag{11}$$

for  $i = 1, \dots, n_{B^2}$ . From Eqs. (3)–(5), the auxiliary signal  $v$  for the current sampling instant  $k$  is

$$v(k) = \sum_{j=1}^{n_{B^1}} b_j^1 u(k-j) - \sum_{j=1}^{n_{A^1}} a_j^1 v(k-j) \tag{12}$$

From Eqs. (10) and (11), one has

$$y(k) = \sum_{i=1}^{n_{B^2}} b_i^2 f(v(k-i)) - \sum_{i=1}^{n_{A^2}} a_i^2 y(k-i) \tag{13}$$

where from Eq. (12)

$$v(k-i) = \sum_{j=1}^{n_{B^1}} b_j^1 u(k-j-i) - \sum_{j=1}^{n_{A^1}} a_j^1 v(k-j-i) \tag{14}$$

for  $i = 1, \dots, n_{B^2}$ . Taking into account Eqs. (11), (13) and (14), it is possible to express the output of the model for the current sampling instant  $k$  as a function of: the model input variable  $u$ , the auxiliary variable  $v$  and the output variable  $y$  at the same previous sampling instants

$$y(k) = \sum_{i=1}^{n_{B^2}} b_i^2 f \left( \sum_{j=1}^{n_{B^1}} b_j^1 u(k-j-i) - \sum_{j=1}^{n_{A^1}} a_j^1 v(k-j-i) \right) - \sum_{i=1}^{n_{A^2}} a_i^2 y(k-i) \tag{15}$$

#### 4 Nonlinear MPC algorithms of Hammerstein–Wiener systems

The Wiener–Hammerstein model is used in MPC for prediction, i.e. to calculate the predicted values of the output variable for the consecutive sampling instants, i.e.  $\hat{y}(k+1|k), \dots, \hat{y}(k+N|k)$ . The prediction for a sampling instant  $k+p$  is calculated at the current instant  $k$  as a sum of the model output supplemented by an estimation of the unmeasured disturbance,  $d(k)$ , [1]. From Eq. (10), one obtains

$$\begin{aligned} \hat{y}(k+p|k) = & \sum_{i=1}^{I_{xf}(p)} b_i^2 x(k-i+p|k) \\ & + \sum_{i=I_{xf}(p)+1}^{n_{B^2}} b_i^2 x(k-i+p) \\ & - \sum_{i=1}^{I_{yf}(p)} a_i^2 \hat{y}(k-i+p|k) \\ & - \sum_{i=I_{yf}(p)+1}^{n_{A^2}} a_i^2 y(k-i+p) + d(k) \end{aligned} \tag{16}$$

for  $p = 1, \dots, N$ , where  $I_{xf}(p) = \min(p, n_{B^2})$ ,  $I_{yf}(p) = \min(p-1, n_{A^2})$ . The symbols  $x(k-1), x(k-2), \dots$  denote the past values of the auxiliary signal between the nonlinear steady-state part and the output linear dynamic part of the model, and their future predicted values are denoted by  $x(k|k), x(k+1|k), \dots$ . Similarly, the symbols  $y(k), y(k-1), \dots$  denote the past values of the output signal (their measurements are available). Using the introduced notation, from Eq. (6) it is obvious that

$$x(k-i+p|k) = f(v(k-i+p|k)) \text{ for } p-i \geq 0 \tag{17a}$$

$$x(k-i+p) = f(v(k-i+p)) \text{ for } p-i < 0 \tag{17b}$$

The prediction equation (16) introduces the integral action into the MPC algorithm which is necessary to compensate for disturbances which affects the process and for the mismatch between the process and its model [1]. It is possible because during prediction calculation not only the model, but also an estimate of the unmeasured disturbance  $d(k)$  acting on the process output is used. The disturbance is assessed as the difference between the real value of the process output variable measured at the current sampling instant, i.e.  $y(k)$ , and the model output. From Eq. (15), one has

$$\begin{aligned}
 d(k) = & y(k) - \sum_{i=1}^{n_{B2}} b_i^2 f \left( \sum_{j=1}^{n_{B1}} b_j^1 u(k-j-i) \right. \\
 & \left. - \sum_{j=1}^{n_{A1}} a_j^1 v(k-j-i) \right) \\
 & + \sum_{i=1}^{n_{A2}} a_i^2 y(k-i) \tag{18}
 \end{aligned}$$

From Eq. (14) one may find the values of the predicted values of the auxiliary signal between the input linear dynamic part and the steady-state nonlinear part. If  $p - i \geq 0$ , one has

$$\begin{aligned}
 v(k-i+p|k) = & \sum_{j=1}^{I_{uf}(i,p)} b_j^1 u(k-j-i+p|k) \\
 & + \sum_{j=I_{uf}(i,p)+1}^{n_{B1}} b_j^1 u(k-j-i+p) \\
 & - \sum_{j=1}^{I_{vf}(i,p)} a_j^1 v(k-j-i+p|k) \\
 & - \sum_{j=I_{vf}(i,p)+1}^{n_{A1}} a_j^1 v(k-j-i+p) \tag{19}
 \end{aligned}$$

where  $I_{uf}(i, p) = \min(p - i, n_{B1})$  and  $I_{vf}(i, p) = \min(p - i, n_{A1})$ , while when  $p - i < 0$ , one obtains

$$\begin{aligned}
 v(k-i+p) = & \sum_{j=1}^{n_{B1}} b_j^1 u(k-j-i+p) \\
 & - \sum_{j=1}^{n_{A1}} a_j^1 v(k-j-i+p) \tag{20}
 \end{aligned}$$

The symbols  $u(k - 1), u(k - 2), \dots$  denote the past values of the manipulated variable (i.e. applied to the process), and their future predicted values (calculated on-line by the MPC algorithm) are denoted by  $u(k|k), u(k + 1|k), \dots$ . Similarly, the symbols  $v(k - 1), v(k - 2), \dots$  denote the past values of the auxiliary signal between the input linear dynamic part and the nonlinear steady-state one, and their predicted values are denoted by  $v(k|k), v(k + 1|k), \dots$

From the prediction equations (16)–(17a) and (19) it is clear that if the Wiener–Hammerstein model is directly used in MPC for prediction calculation, the obtained predicted values of the output variable over the prediction horizon, i.e. the signals  $\hat{y}(k + 1|k), \dots, \hat{y}(k + N|k)$ , are nonlinear functions of the decision variables of MPC, i.e. the calculated on-line future control increments  $\Delta u(k|k), \dots, \Delta u(k + N_u|k)$ . A direct consequence of this fact is that the MPC optimisation problem (2) becomes a constrained nonlinear task which must be solved on-line in real time. It means that a nonlinear solver must be used, and the optimisation problem is likely to be computationally demanding. Such a truly nonlinear MPC strategy is named the MPC algorithm with nonlinear optimisation (MPC-NO). In the following part of the article two nonlinear MPC algorithms with on-line linearisation (performed by means of different methods) are detailed which do not require solving nonlinear optimisation problems in real time, and relatively computationally not demanding quadratic optimisation is used. Provided that  $\lambda > 0$ , the resulting quadratic optimisation tasks have only one global solution.

#### 4.1 Nonlinear MPC algorithm with on-line simplified model linearisation

The simplest approach to nonlinear MPC of dynamic systems represented by the Wiener–Hammerstein model is to use for prediction a linear time-varying model obtained by linearisation of the nonlinear steady-state part of the model. For this purpose the specific serial structure of the model shown in Fig. 1 is exploited. Taking into account Eq. (6), the gain of the nonlinear block for the current operating point is

$$K(k) = \frac{dx(k)}{dv(k)} = \frac{df(v(k))}{dv(k)} \tag{21}$$

The output signal of the nonlinear steady-state part can be simply calculated for the current operating point from  $x(k) = K(k)v(k)$ . Hence, from Eq. (7), the nonlinear steady-state part and the output linear dynamic part of the model may be described by

$$A^2(q^{-1})y(k) = B^2(q^{-1})K(k)v(k)$$

Next, taking into account the input linear dynamic part defined by Eq. (3), the linearised model becomes

$$A^1(q^{-1})A^2(q^{-1})y(k) = B^1(q^{-1})B^2(q^{-1})K(k)u(k)$$

The linearised model may be compactly defined by

$$A(q^{-1})y(k) = K(k)B(q^{-1})u(k) \tag{22}$$

where the polynomials in the backward shift operator are

$$\begin{aligned} A(q^{-1}) &= A^1(q^{-1})A^2(q^{-1}) \\ &= 1 + a_1q^{-1} + \dots + a_{n_A}q^{-n_A} \\ B(q^{-1}) &= B^1(q^{-1})B^2(q^{-1}) \\ &= b_1q^{-1} + \dots + b_{n_B}q^{-n_B} \end{aligned}$$

the order of dynamics is defined by  $n_A = n_{A^1} + n_{A^2}$ ,  $n_B = n_{B^1} + n_{B^2}$  and the coefficients are

$$\begin{aligned} a_1 &= a_1^2 + a_1^1 \\ a_2 &= a_2^2 + a_1^1a_1^2 + a_2^1 \\ a_3 &= a_3^2 + a_1^1a_2^2 + a_2^1a_1^2 + a_3^1 \\ a_4 &= a_4^2 + a_1^1a_3^2 + a_2^1a_2^2 + a_3^1a_1^2 + a_4^1 \\ a_5 &= a_5^2 + a_1^1a_4^2 + a_2^1a_3^2 + a_3^1a_2^2 + a_4^1a_1^2 + a_5^1 \\ &\vdots \\ b_2 &= b_1^1b_1^2 \\ b_3 &= b_1^1b_2^2 + b_2^1b_1^2 \\ b_4 &= b_1^1b_3^2 + b_2^1b_2^2 + b_3^1b_1^2 \\ b_5 &= b_1^1b_4^2 + b_2^1b_3^2 + b_3^1b_2^2 + b_4^1b_1^2 \\ &\vdots \end{aligned}$$

which can be shortly expressed by

$$\begin{aligned} a_i &= a_i^1 + a_i^2 + \sum_{j=1}^{i-1} a_j^1a_{i-j}^2, \quad i = 1, \dots, n_A \\ b_i &= \sum_{j=1}^{i-1} b_j^1b_{i-j}^2, \quad i = 1, \dots, n_B \end{aligned} \tag{23}$$

Because the nonlinear Wiener–Hammerstein system is approximated for the current sampling instant  $k$  by the linear time-varying model (22), the predicted trajectory of the output variable over the prediction horizon

$$\hat{y}(k) = [\hat{y}(k+1|k) \dots \hat{y}(k+N|k)]^T \tag{24}$$

is a linear function of the decision variables of MPC, i.e. the calculated control increments (1), as derived in [12, 18]. The prediction equation in the vector-matrix notation is

$$\hat{y}(k) = G(k)\Delta u(k) + y^0(k) \tag{25}$$

where the matrix  $G(k)$ , of dimensionality  $N \times N_u$ , contains step-response coefficients of the local linear approximation of the nonlinear model. Using Eq. (22), it follows that

$$G(k) = K(k)G \tag{26}$$

where the constant matrix

$$G = \begin{bmatrix} s_1 & 0 & \dots & 0 \\ s_2 & s_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ s_N & s_{N-1} & \dots & s_{N-N_u+1} \end{bmatrix}$$

consists of step-response coefficients of two linear dynamic parts of the Wiener–Hammerstein model connected in series, i.e. the dynamic system described by  $A(q^{-1})y(k) = B(q^{-1})u(k)$ . Such step-response coefficients are calculated from the quantities defined by Eq. (23) in the following way

$$s_p = \sum_{i=1}^{\min(p, n_B)} b_i - \sum_{i=1}^{\min(j-1, n_A)} a_i s_{j-i}$$



for  $p = 1, \dots, N$ . Using Eq. (26), the prediction equation (25) becomes

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

In Eq. (27) the predicted output trajectory,  $\hat{y}(k)$ , is a sum of the forced trajectory  $K(k)G\Delta u(k)$ , which depends only on the future control increments (1), and the free trajectory vector  $y^0(k) = [y^0(k + 1|k) \dots y^0(k + N|k)]^T$ , which entirely depends on the past, i.e. on the values of the manipulated variable applied to the process before the instant  $k$ . The free trajectory is calculated from the nonlinear Wiener–Hammerstein model. From Eq. (16), one has

$$\begin{aligned} y^0(k + p|k) &= \sum_{i=1}^{I_{xf}(p)} b_i^2 x^0(k - i + p|k) \\ &+ \sum_{i=I_{xf}(p)+1}^{n_{B2}} b_i^2 x(k - i + p) \\ &- \sum_{i=1}^{I_{yf}(p)} a_i^2 y^0(k - i + p|k) \\ &- \sum_{i=I_{yf}(p)+1}^{n_{A2}} a_i^2 y(k - i + p) + d(k) \end{aligned} \tag{28}$$

where  $p = 1, \dots, N$ ,  $x^0(k + p|k)$  denotes the auxiliary signal between the nonlinear steady-state part and the output linear dynamic one predicted for the sampling instant  $k + p$  calculated at the instant  $k$ . From Eq. (6)

$$x^0(k - i + p|k) = f(v^0(k - i + p|k)) \tag{29}$$

where from Eq. (19) the auxiliary signal between the input linear dynamic part and the nonlinear steady-state one predicted for the sampling instant  $k + p$  calculated at the instant  $k$  is

$$\begin{aligned} v^0(k - i + p|k) &= \sum_{j=1}^{I_{uf}(i,p)} b_j^1 u(k - 1) \\ &+ \sum_{j=I_{uf}(i,p)+1}^{n_{B1}} b_j^1 u(k - j - i + p) \end{aligned}$$

$$\begin{aligned} &- \sum_{j=1}^{I_{vf}(i,p)} a_j^1 v^0(k - j - i + p|k) \\ &- \sum_{j=I_{vf}(i,p)+1}^{n_{A1}} a_j^1 v(k - j - i + p) \end{aligned} \tag{30}$$

and the past quantities  $v(k - i + p)$  are found from Eq. (20). As the predicted output trajectory (27) is a linear function of the future control increments  $\Delta u(k)$ , the MPC optimisation problem (2) becomes the quadratic programming task

$$\begin{aligned} &\min_{\Delta u(k)} \left\{ \|y^{sp}(k) - y^0(k) - K(k)G\Delta u(k)\|^2 + \|\Delta u(k)\|_{\Lambda}^2 \right\} \\ &\text{subject to} \\ &u^{\min} \leq J\Delta u(k) + u(k - 1) \leq u^{\max} \\ &-\Delta u^{\max} \leq \Delta u(k) \leq \Delta u^{\max} \\ &y^{\min} \leq y^0(k) + K(k)G\Delta u(k) \leq y^{\max} \end{aligned} \tag{31}$$

where  $\|x\|^2 = x^T x$  and  $\|x\|_{\Lambda}^2 = x^T \Lambda x$ , the set-point trajectory vector  $y^{sp}(k) = [y^{sp}(k + 1|k) \dots y^{sp}(k + N|k)]^T$  and the vectors of output constraints, i.e.  $y^{\min} = [y^{\min} \dots y^{\min}]^T$ ,  $y^{\max} = [y^{\max} \dots y^{\max}]^T$ , are of length  $N$ . The vectors of input constraints, i.e.  $u^{\min} = [u^{\min} \dots u^{\min}]^T$ ,  $u^{\max} = [u^{\max} \dots u^{\max}]^T$ ,  $\Delta u^{\max} = [\Delta u^{\max} \dots \Delta u^{\max}]^T$  and the vector  $u(k - 1) = [u(k - 1) \dots u(k - 1)]^T$  are of length  $N_u$ , the matrices  $\Lambda = \text{diag}(\lambda, \dots, \lambda)$  and

$$J = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix} \tag{32}$$

are of dimensionality  $N_u \times N_u$ .

The described MPC strategy is referred to as the MPC algorithm with nonlinear prediction and simplified linearisation (MPC-NPSL). It is because, at each sampling instant on-line, the nonlinear Wiener–Hammerstein model is used for prediction of the nonlinear free trajectory and the same model is used to find a linear approximation of the model for the current operating point. Linearisation is carried out in a simplified way, because the linear dynamic blocks are multiplied by the current gain of the nonlinear steady-state one. At each sampling instant  $k$  of the

algorithm the gain  $K(k)$  of the steady-state part of the model for the current operating point is estimated from Eq. (21) and the nonlinear free trajectory  $\mathbf{y}^0(k)$  is found from Eqs. (28)–(30), (20), and the unmeasured disturbance is estimated from Eq. (18). The MPC-NPSL quadratic optimisation problem (31) is solved to find the future control increments  $\Delta \mathbf{u}(k)$ , the first element of which is applied to the process, i.e. set  $u(k) = \Delta u(k|k) + u(k-1)$ . The MPC-NPSL quadratic optimisation problem has only one unique global solution provided that  $\Lambda > 0$  (i.e.  $\lambda > 0$ ).

The same approach may be also used for the Wiener model [23]. It is also possible to use the linearised model for calculation of the free trajectory [22]. It is necessary to stress the fact that the MPC-NPSL algorithm does not require an inverse model of the nonlinear steady-state part of the model, as it is necessary in the MPC algorithms with on-line model linearisation for cascade Wiener [19–21] and Hammerstein [15–17] systems, respectively.

#### 4.2 Nonlinear MPC algorithm with on-line predicted trajectory linearisation

Although the time-varying linear approximation of the Wiener–Hammerstein system is calculated for the current operating conditions of the process, the same linearised model is used for the calculation of the output predictions for the whole prediction horizon. When the horizon is long and the operating point is changed significantly, such a linearised model is likely to describe properties of the real nonlinear process not precisely, i.e. the mismatch between the model prediction and behaviour of the real process may be significant. It may result in not acceptable quality of control. A conceptually better method is to calculate not a linear approximation of the model and next use it for finding the predicted trajectory, but to directly find a linear approximation of the nonlinear predicted trajectory defined on the prediction horizon,  $\hat{\mathbf{y}}(k)$ , Eq. (24). It is important to stress the fact that linearisation is not carried out for the current operating point of the process, as in the MPC-NPSL algorithm, but along an assumed future sequence of the manipulated variable  $\mathbf{u}^{\text{traj}}(k) = [u^{\text{traj}}(k|k) \dots u^{\text{traj}}(k + N_u - 1|k)]^T$ . Using the Taylor's series expansion method, a linear approximation of the nonlinear output trajectory  $\hat{\mathbf{y}}(k)$  along the input trajectory  $\mathbf{u}^{\text{traj}}(k)$  is

$$\hat{\mathbf{y}}(k) = \hat{\mathbf{y}}^{\text{traj}}(k) + \mathbf{H}(k) \left( \mathbf{u}(k) - \mathbf{u}^{\text{traj}}(k) \right) \quad (33)$$

where the output trajectory  $\hat{\mathbf{y}}^{\text{traj}}(k)$  describes the predictions for the input trajectory  $\mathbf{u}^{\text{traj}}(k)$  and  $\mathbf{H}(k)$  is the matrix of the derivatives of the predicted output trajectory with respect to the future values of the control signal. The obtained approximation formula (33) is really a linear function of the future values of the control variable over the control horizon, i.e.  $\mathbf{u}(k) = [u(k|k) \dots u(k + N_u - 1|k)]^T$ . The main issue is the choice of the future input trajectory  $\mathbf{u}^{\text{traj}}(k)$ , along which linearisation is performed. Ideally, it should be close to the trajectory corresponding to the optimal control increments (1), which is not known at the sampling instant  $k$ , because this is the decision variable vector in MPC, calculated at the same instant. A possible solution is to use for linearisation the trajectory determined by the most recent value of the manipulated variable, i.e. the value applied to the process at the previous sampling instant,  $\mathbf{u}^{\text{traj}}(k) = [u(k-1) \dots u(k-1)]^T$ . A better approach is to repeat trajectory linearisation and calculation of the future control policy a few times at each sampling instants in the internal iterations  $t = 1, \dots, t^{\text{max}}$ . In the current internal iteration  $t$  the predicted nonlinear output trajectory  $\hat{\mathbf{y}}^t(k) = [\hat{y}^t(k+1|k) \dots \hat{y}^t(k+N|k)]^T$  is linearised along the input trajectory found in the previous internal iteration ( $t-1$ ), i.e.  $\mathbf{u}^{t-1}(k) = [u^{t-1}(k|k) \dots u^{t-1}(k+N_u-1|k)]^T$ . Analogously to the case when the trajectory is linearised along some assumed input trajectory  $\mathbf{u}^{\text{traj}}(k)$ , which results in the approximation (33), using the Taylor's series expansion method, a linear approximation of the nonlinear output trajectory  $\hat{\mathbf{y}}^t(k)$  along the input trajectory  $\mathbf{u}^{t-1}(k)$  is

$$\hat{\mathbf{y}}^t(k) = \hat{\mathbf{y}}^{t-1}(k) + \mathbf{H}^t(k) \left( \mathbf{u}^t(k) - \mathbf{u}^{t-1}(k) \right) \quad (34)$$

where the output trajectory corresponding to the input trajectory calculated at the previous internal iteration,  $\mathbf{u}^{t-1}(k)$ , is denoted by  $\hat{\mathbf{y}}^{t-1}(k) = [\hat{y}^{t-1}(k+1|k) \dots \hat{y}^{t-1}(k+N|k)]^T$  and  $\mathbf{H}^t(k)$  is the matrix of the derivatives of the predicted output trajectory with respect to the future values of the control signal. The matrix is calculated for the input and output trajectories found in the previous internal iteration,  $t-1$ , because no new information is available. It is of dimensionality  $N \times N_u$  and has the structure



$$\begin{aligned}
 \mathbf{H}^t(k) &= \left. \frac{d\hat{\mathbf{y}}(k)}{d\mathbf{u}(k)} \right|_{\substack{\hat{\mathbf{y}}(k)=\hat{\mathbf{y}}^{t-1}(k) \\ \mathbf{u}(k)=\mathbf{u}^{t-1}(k)}} \\
 &= \begin{bmatrix} \frac{\partial \hat{y}^{t-1}(k+1|k)}{\partial u^{t-1}(k|k)} & \cdots & \frac{\partial \hat{y}^{t-1}(k+1|k)}{\partial u^{t-1}(k+N_u-1|k)} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}^{t-1}(k+N|k)}{\partial u^{t-1}(k|k)} & \cdots & \frac{\partial \hat{y}^{t-1}(k+N|k)}{\partial u^{t-1}(k+N_u-1|k)} \end{bmatrix} \quad (35)
 \end{aligned}$$

where  $p = 1, \dots, N$  and  $r = 0, \dots, N_u - 1$ . Equation (34) is really a linear function of the future values of the control variable over the control horizon calculated at the current internal iteration  $t$ , i.e.  $\mathbf{u}^t(k) = [u^t(k|k) \dots u^t(k + N_u - 1|k)]^T$ . It is possible to express the linear approximation of the nonlinear output trajectory as a function of the increments of the manipulated variable calculated for the current sampling instant  $k$  and in the current internal iteration  $t$ , i.e. the trajectory  $\Delta \mathbf{u}^t(k) = [\Delta u^t(k|k) \dots \Delta u^t(k + N_u - 1|k)]^T$  corresponding to the vector (1). It is necessary because future increments of the manipulated variable, not its future values, are minimised in the second part of the MPC cost function  $J(k)$ . Taking into account that  $\mathbf{u}^t(k) = \mathbf{J}\Delta \mathbf{u}^t(k) + \mathbf{u}(k - 1)$ , where the matrix  $\mathbf{J}$  of dimensionality  $N_u \times N_u$  is defined by Eq. (32) and the vector  $\mathbf{u}(k - 1) = [u(k - 1) \dots u(k - 1)]^T$  is of length  $N_u$ , the linear approximation of the nonlinear predicted output trajectory (34) is

$$\begin{aligned}
 \hat{\mathbf{y}}^t(k) &= \mathbf{H}^t(k)\mathbf{J}\Delta \mathbf{u}^t(k) + \hat{\mathbf{y}}^{t-1}(k) \\
 &\quad + \mathbf{H}^t(k)\left(\mathbf{u}(k - 1) - \mathbf{u}^{t-1}(k)\right) \quad (36)
 \end{aligned}$$

It is easy to note that the obtained approximation formula (36) is really a linear function of the future increments of the control variable over the control horizon, i.e.  $\Delta \mathbf{u}^t(k)$ , which is calculated for the current sampling instant  $k$  and in the current internal iteration  $t$ . The vector  $\mathbf{u}^{t-1}(k)$  is known, the matrix  $\mathbf{J}$  is fixed, whereas the vector  $\hat{\mathbf{y}}^{t-1}(k)$  and the matrix  $\mathbf{H}^t(k)$  may be determined from the Wiener–Hammerstein model of the process. From Eq. (16), the predicted value of the output for the sampling instant  $k + p$  calculated at the current instant  $k$  and at the current internal iteration  $t$  is

$$\hat{y}^t(k + p|k) = \sum_{i=1}^{I_{xf}(p)} b_i^2 x^t(k - i + p|k)$$

$$\begin{aligned}
 &+ \sum_{i=I_{xf}(p)+1}^{n_{B2}} b_i^2 x(k - i + p) \\
 &- \sum_{i=1}^{I_{yf}(p)} a_i^2 \hat{y}^t(k - i + p|k) \\
 &- \sum_{i=I_{yf}(p)+1}^{n_{A2}} a_i^2 y(k - i + p) + d(k) \quad (37)
 \end{aligned}$$

where from Eq. (6), one has

$$x^t(k - i + p|k) = f(v^t(k - i + p|k)) \quad (38)$$

for  $p - i \geq 0$ . From Eq. (19), the predicted signals between the input linear dynamic part and the nonlinear steady-state one are

$$\begin{aligned}
 v^t(k - i + p|k) &= \sum_{j=1}^{I_{uf}(i,p)} b_j^1 u^t(k - j - i + p|k) \\
 &+ \sum_{j=I_{uf}(i,p)+1}^{n_{B1}} b_j^1 u(k - j - i + p) \\
 &- \sum_{j=1}^{I_{vf}(i,p)} a_j^1 v^t(k - j - i + p|k) \\
 &- \sum_{j=I_{vf}(i,p)+1}^{n_{A1}} a_j^1 v(k - j - i + p) \quad (39)
 \end{aligned}$$

and the past quantities  $v(k - i + p)$  are calculated from Eq. (20). The entries of the matrix (35), i.e. the derivatives of the predicted trajectory  $\hat{\mathbf{y}}^{t-1}(k)$  for the previous internal iteration,  $t - 1$ , with respect to the future control scenario  $\Delta \mathbf{u}^{t-1}(k)$  are found by differentiating Eq. (37), which gives

$$\begin{aligned}
 \frac{\partial y^{t-1}(k + p|k)}{\partial u^{t-1}(k + r|k)} &= \sum_{i=1}^{I_{xf}(p)} b_i^2 \frac{\partial x^{t-1}(k - i + p|k)}{\partial u^{t-1}(k + r|k)} \\
 &- \sum_{i=1}^{I_{yf}(p)} a_i^2 \frac{\partial y^{t-1}(k - i + p)}{\partial u^{t-1}(k + r|k)} \quad (40)
 \end{aligned}$$

The first partial derivative in the right side of Eq. (40) depends on the specific type of the nonlinear steady-state block, described by the function  $f$

$$\frac{\partial x^{t-1}(k-i+p|k)}{\partial u^{t-1}(k+r|k)} = \frac{df(v^{t-1}(k-i+p|k))}{dv^{t-1}(k-i+p)} \times \frac{\partial v^{t-1}(k-i+p|k)}{\partial u^{t-1}(k+r|k)} \tag{41}$$

where from Eq. (39), the derivatives of the predicted signals between the input linear dynamic part and the nonlinear steady-state one are

$$\frac{\partial v^{t-1}(k-i+p|k)}{\partial u^{t-1}(k+r|k)} = \sum_{j=1}^{I_{uf}(i,p)} b_j^1 \frac{\partial u^{t-1}(k-j-i+p|k)}{\partial u^{t-1}(k+r|k)} - \sum_{j=1}^{I_{vf}(i,p)} a_j^1 \frac{\partial v^{t-1}(k-j-i+p|k)}{\partial u^{t-1}(k+r|k)} \tag{42}$$

Because  $u(k+p|k) = u(k+N_u-1|k)$  for  $p \geq N_u$ , the first partial derivatives in the right side of Eq. (42) can only have two values

$$\frac{\partial u^{t-1}(k+p|k)}{\partial u^{t-1}(k+r|k)} = \begin{cases} 1 & \text{if } p = r \text{ or } (p > r \text{ and } r = N_u - 1) \\ 0 & \text{otherwise} \end{cases} \tag{43}$$

The second partial derivatives in Eqs. (40) and (42) must be calculated recursively. One may notice that

$$\frac{\partial \hat{y}^{t-1}(k+p|k)}{\partial u^{t-1}(k+r|k)} = 0 \tag{44}$$

for  $r \geq p-1$  and

$$\frac{\partial v^{t-1}(k+p|k)}{\partial u^{t-1}(k+r|k)} = 0 \tag{45}$$

for  $r \geq p$ .

According to Eq. (36), linearisation makes it possible to express the predicted output trajectory  $\hat{y}^t(k)$  as a linear function of the future control increments  $\Delta u^t(k)$  calculated for the current sampling instant  $k$  and at the internal iteration  $t$ . Hence, the MPC optimisation problem (2) becomes the following quadratic programming task

$$\min_{\Delta u^t(k)} \left\{ \left\| \mathbf{y}^{sp}(k) - \mathbf{H}^t(k) \mathbf{J} \Delta u^t(k) - \hat{\mathbf{y}}^{t-1}(k) \right. \right. \\ \left. \left. - \mathbf{H}^t(k) (\mathbf{u}(k-1) - \mathbf{u}^{t-1}(k)) \right\|^2 + \left\| \Delta u^t(k) \right\|_{\Lambda}^2 \right\}$$

subject to

$$\mathbf{u}^{\min} \leq \mathbf{J} \Delta u^t(k) + \mathbf{u}(k-1) \leq \mathbf{u}^{\max} \\ - \Delta \mathbf{u}^{\max} \leq \Delta u^t(k) \leq \Delta \mathbf{u}^{\max} \\ \mathbf{y}^{\min} \leq \mathbf{H}^t(k) \mathbf{J} \Delta u^t(k) + \hat{\mathbf{y}}^{t-1}(k) \\ + \mathbf{H}^t(k) (\mathbf{u}(k-1) - \mathbf{u}^{t-1}(k)) \leq \mathbf{y}^{\max} \tag{46}$$

The obtained quadratic optimisation problem, analogously to that used in the MPC-NPSL algorithm [(Eq. (31)], has only one unique global solution provided that  $\Lambda > 0$  (i.e.  $\lambda > 0$ ).

The described MPC strategy is referred to as the MPC algorithm with nonlinear prediction and linearisation along the predicted trajectory (MPC-NPLPT). At each sampling instant  $k$  of the algorithm the unmeasured disturbance is estimated from Eq. (18) using the Wiener–Hammerstein model of the process. Next, the internal iteration is initialised ( $t = 1$ ): an initial guess of the future input trajectory is assumed, e.g. using the value of the manipulated variable calculated and applied to the process at the previous sampling instant, i.e.  $\mathbf{u}^0(k) = [u(k-1) \dots u(k-1)]^T$ , or the last  $N_u-1$  elements of the optimal trajectory calculated at the previous instant and not applied to the process, i.e. the trajectory

$$\mathbf{u}^0(k) = \begin{bmatrix} u^0(k|k) \\ \vdots \\ u^0(k+N_u-3|k) \\ u^0(k+N_u-2|k) \\ u^0(k+N_u-1|k) \end{bmatrix} = \begin{bmatrix} u(k|k-1) \\ \vdots \\ u(k+N_u-3|k-1) \\ u(k+N_u-2|k-1) \\ u(k+N_u-2|k-1) \end{bmatrix} \tag{47}$$

For the initial future input trajectory  $\mathbf{u}^0(k)$ , the future output trajectory  $\hat{\mathbf{y}}^0(k)$  is calculated from Eqs. (17b), (19), (37), (38) and (39). The matrix  $\mathbf{H}^t(k)$ , which consists of derivatives of the predicted output trajectory with respect to the future control policy, specified by Eq. (35), is calculated using Eqs. (40)–(42) and the conditions (43)–(44). The MPC-NPLPT quadratic optimisation problem (46) is solved to find the future control

increments  $\Delta \mathbf{u}^1(k)$  in the first internal iteration. If the condition

$$\sum_{p=0}^{N_0} \|y^{SP}(k-p) - y(k-p)\|^2 \geq \delta_y \tag{48}$$

is true, the internal iterations are continued ( $t = 2, \dots, t_{max}$ ). The tuning parameters  $N_0$  and  $\delta_y$  are chosen experimentally. In the internal iteration  $t$  the predicted output trajectory  $\hat{y}^{t-1}(k)$  corresponding to the future input trajectory  $\mathbf{u}^{t-1}(k) = \mathbf{J} \Delta \mathbf{u}^{t-1}(k) + \mathbf{u}(k-1)$  and the matrix  $\mathbf{H}^t(k)$  are calculated (in the same way it is done in the first internal iteration). The MPC-NPLPT quadratic optimisation problem (46) is solved to find the future control increments  $\Delta \mathbf{u}^t(k)$ . If the difference between future control increments calculated in the consecutive internal iterations is small, it means when

$$\|\Delta \mathbf{u}^t(k) - \Delta \mathbf{u}^{t-1}(k)\|^2 < \delta_u \tag{49}$$

or  $t > t_{max}$ , internal iterations are terminated. Otherwise, the next internal iteration is started ( $t := t + 1$ ), the nonlinear output trajectory is linearised and the quadratic optimisation problem is solved. The tuning parameter  $\delta_u$  is chosen experimentally. Having completed the internal iterations, the first element of the obtained future control policy is applied to the process, i.e. set  $u(k) = \Delta u^t(k|k) + u(k-1)$ .

The MPC-NPLPT algorithm for Hammerstein and Wiener systems with a specific nonlinear part represented by a neural network is discussed in [12,23], and the algorithm for a general Hammerstein–Wiener model is described in [26].

### 5 Simulation results

The considered process is a heat exchanger [30], whose control system is depicted in Fig. 2. Cold water (the input stream) is heated by a hot steam, and the process produces hot water (the output stream). The objective of the temperature controller is to calculate the value of the manipulated variable (process input),  $u$ , which is typically a electric signal, in such a way that temperature of hot water, which is the controlled variable (process output),  $y$ , follows changes of its set point. The heat exchanger may be modelled as a

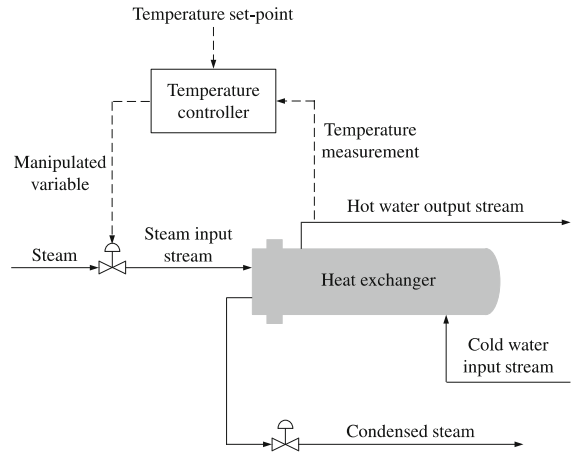


Fig. 2 The control system of the heat exchanger

Wiener–Hammerstein structure shown in Fig. 1. The first (input) linear dynamic part of the model represents the dynamic subsystem between the manipulated variable and the valve position,  $v$ , and it is defined by the second-order equation

$$(1 - 1.5714q^{-1} + 0.6873q^{-2})v(k) = (0.0616q^{-1} + 0.0543q^{-2})u(k) \tag{50}$$

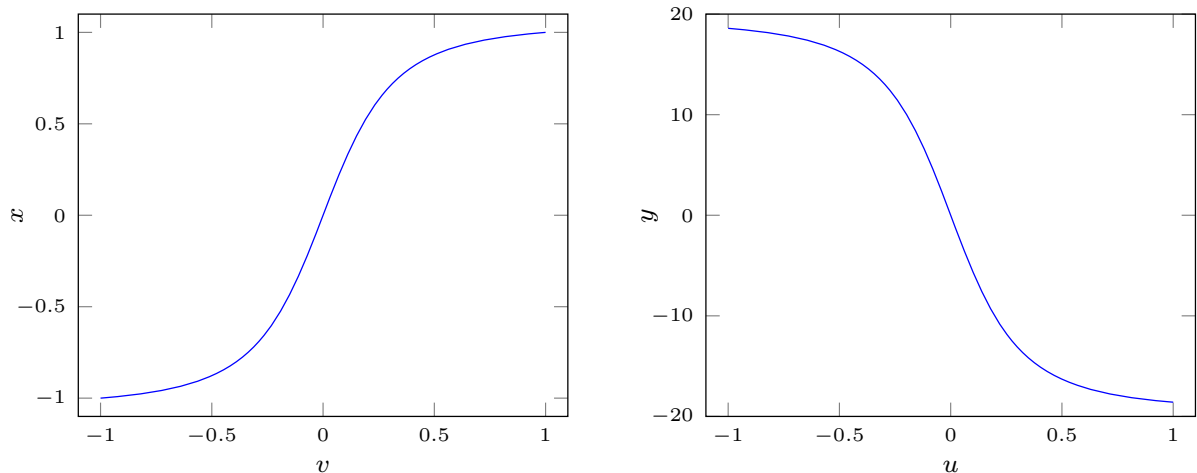
i.e.  $a_1^1 = -1.5714$ ,  $a_2^1 = 0.6873$ ,  $b_1^1 = 0.0616$ ,  $b_2^1 = 0.0543$ ,  $n_{A1} = n_{B1} = 2$ . The nonlinear steady-state block represents the relation between the valve position and the flowrate of the input stream actually delivered to the heat exchanger,  $x$ , and it is defined by the nonlinear function

$$x(k) = f(v(k)) = \frac{v(k)}{\sqrt{0.1 + 0.9v^2(k)}} \tag{51}$$

Finally, the second (output) linear dynamic block represents the dynamic subsystem between the steam input stream and the temperature of the hot water and it is defined by the second-order equation

$$(1 - 1.7608q^{-1} + 0.7661q^{-2})v(k) = (-5.7715q^{-1} + 5.673q^{-2})u(k) \tag{52}$$

i.e.  $a_1^2 = -1.7608$ ,  $a_2^2 = 0.7661$ ,  $b_1^2 = -5.7715$ ,  $b_2^2 = 5.673$ ,  $n_{A2} = n_{B2} = 2$ . Figure 3 shows the characteristics  $x = f(v)$  of the nonlinear steady-state



**Fig. 3** The characteristics  $x = f(v)$  of the nonlinear steady-state block and the characteristics  $y = f(u)$  of the whole Wiener–Hammerstein system

block and the characteristics  $y = f(u)$  of the whole Wiener–Hammerstein system.

In order to demonstrate advantages of the nonlinear MPC algorithms discussed in this work, the following MPC algorithms are compared:

- (a) The classical linear MPC algorithm (the generalised predictive control [1]). For prediction and optimisation of the future control increments, a linear model corresponding to the nominal operating point  $u = v = x = y = 0$  is used. The order of dynamics of the model is  $n_A = n_{A^1} + n_{A^2} = 4$ ,  $n_B = n_{B^1} + n_{B^2} = 4$ , and from Eq. (22), it has the form

$$A(q^{-1})y(k) = K\mathbf{B}(q^{-1})u(k)$$

Model coefficients are calculated from Eqs. (23), (50) and (52), which gives

$$A(q^{-1}) = 1 - 3.3322q^{-1} + 4.2203q^{-2} - 2.4140q^{-3} + 0.5265q^{-4}$$

$$\mathbf{B}(q^{-1}) = -0.3555q^{-2} + 0.0361q^{-3} + 0.3080q^{-4}$$

From Eq. (51), the constant gain of the nonlinear steady-state part of the model is

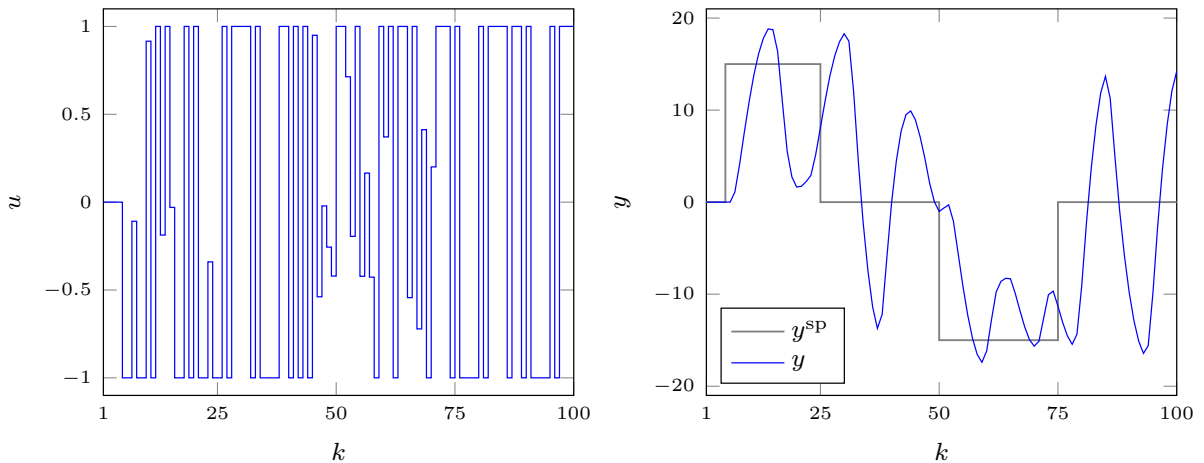
$$\begin{aligned} K &= \frac{dx}{dv} = \left. \frac{dx}{dv} \right|_{x=0, v=0} \\ &= \left. \frac{df(v)}{dv} \right|_{v=0} = 0.1^{-0.5} = 3.1623 \end{aligned}$$

- (b) The described MPC algorithm with on-line simplified linearisation (MPC-NPSL). The nonlinear Wiener–Hammerstein model is used to update the current gain of the parameter-varying linear model specified by Eq. (22). For the given steady-state nonlinear part of the model defined by Eq. (51), the gain of the nonlinear block for the current operating point, defined as the derivative of the auxiliary signal  $x$  of the model with respect to the auxiliary signal  $v$  (Eq. 21), is calculated as

$$\begin{aligned} K(k) &= \frac{dx(k)}{dv(k)} = \frac{df(v(k))}{dv(k)} = (0.1 + 0.9v^2(k))^{-0.5} \\ &\quad - 0.9v^2(k)(0.1 + 0.9v^2(k))^{-1.5} \end{aligned} \quad (53)$$

where the value  $v(k)$  defines the current operating point.

- (c) The MPC algorithm with on-line trajectory linearisation (MPC-NPLPT). For initialisation of linearisation  $N_u - 1$  elements of the future control sequence calculated at the previous sampling instant are used, and the trajectory  $\mathbf{u}^0(k)$  is defined by Eq. (47). The derivative of the predicted auxiliary signal  $x$  of the model with respect to the auxiliary signal  $v$ , predicted for the sampling instant  $k - i + p$  at the current instant  $k$  and in the internal iteration  $t$  (necessary for trajectory linearisation in Eq. (41)), using Eq. (51), is calculated from



**Fig. 4** Simulation results of the linear MPC algorithm;  $\lambda = 1$

$$\begin{aligned} & \frac{df(v^{t-1}(k-i+p|k))}{dv^{t-1}(k-i+p)} \\ &= (0.1 + 0.9(v^{t-1}(k-i+p|k))^2)^{-0.5} \\ & \quad - 0.9(v^{t-1}(k-i+p|k))^2 \\ & \quad \times (0.1 + 0.9(v^{t-1}(k-i+p|k))^2)^{-1.5} \end{aligned}$$

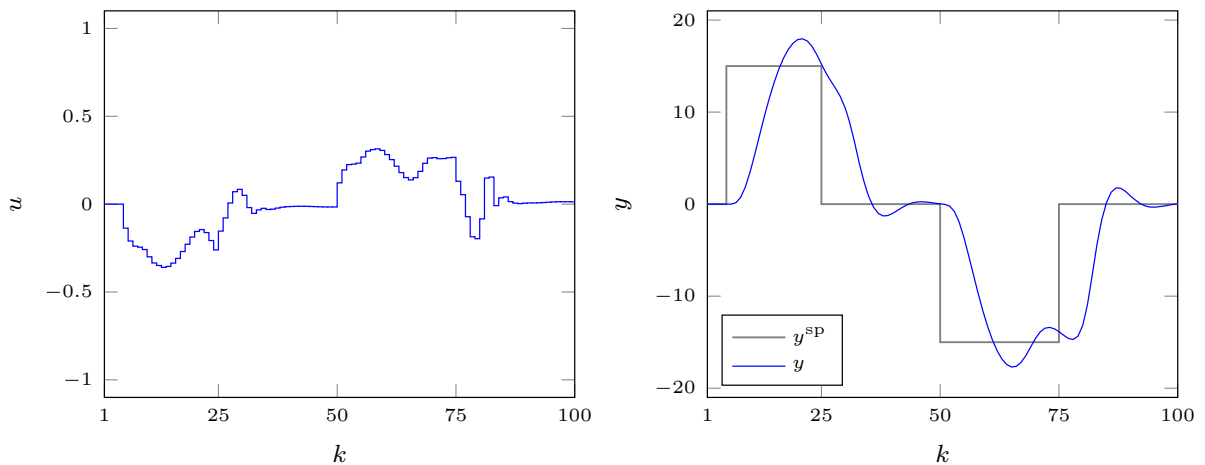
- (d) The “ideal” MPC algorithm with nonlinear optimisation (MPC-NO) in which the full nonlinear Wiener–Hammerstein model is used for prediction and optimisation of the future control policy [Eqs. (16)–(20)].

In the nonlinear MPC algorithms, i.e. in MPC-NPSL, MPC-NPLPT and MPC-NO, the same nonlinear Hammerstein–Wiener model is used, although in a different way. At each sampling instant of the linear MPC strategy and in the MPC-NPSL algorithm a constrained quadratic optimisation problem is solved on-line, in the MPC-NPLPT maximally  $t^{\max}$  such problems are necessary, whereas in the MPC-NO strategy a nonlinear optimisation task is solved. All algorithms are implemented in MATLAB. The active-set algorithm is used for quadratic optimisation, and the sequential quadratic programming (SQP) method is used for nonlinear optimisation [35], both with their default parameters. The same dynamic model described by Eqs. (50)–(52) is used as the simulated system. Parameters of all compared MPC algorithms are the same:  $N = 10$ ,  $N_u = 3$ , typically  $\lambda = 1$ , the constraints imposed on the value of the manipulated

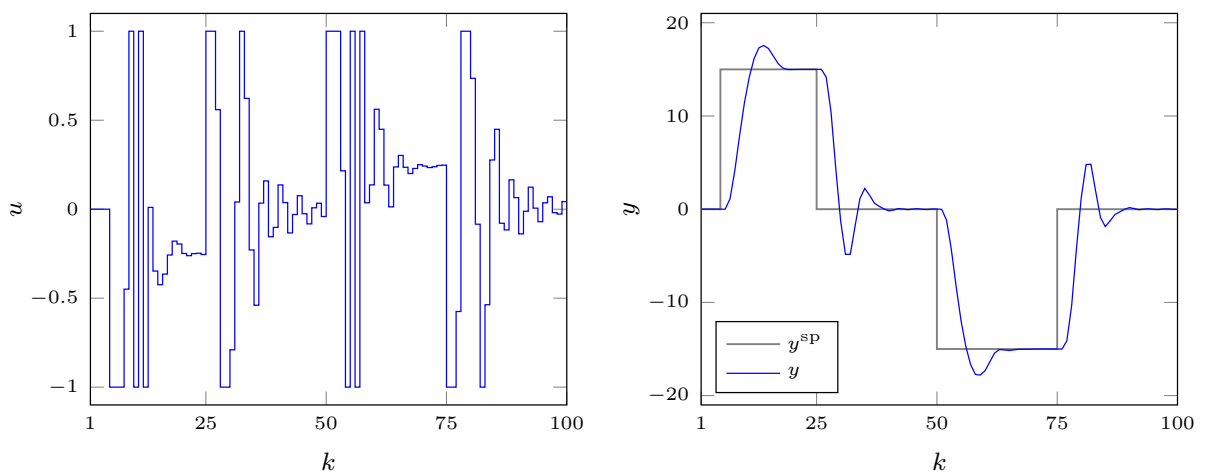
variable are:  $u^{\min} = -1$ ,  $u^{\max} = 1$ . If necessary, the additional constraints imposed on the rate of change of the manipulated variable are also taken into account,  $\Delta u^{\max} = 0.25$ .

At first, the linear MPC algorithm is simulated. Figure 4 depicts the obtained simulation results. The set-point trajectory consists of four fast step changes. Because there is a significant mismatch between the nonlinear process and the linear model used for prediction, the linear MPC algorithm does not work, and it is unable to steer the process in such a way that the controlled variable follows the changes of the set-point trajectory with no steady-state error. Conversely, there is practically no steady-state, and the output variable of the process oscillates. In order to reduce the oscillations, the penalty factor  $\lambda$  must be increased to the value 8000 (or more). Figure 5 shows simulation results. Increasing the value of  $\lambda$  makes it possible to obtain stable control, but changes of the manipulated variable are very slow which leads to unacceptably slow trajectory tracking of the controlled output.

Next, the MPC-NPSL algorithm with simplified linearisation is considered. The linear MPC algorithm uses for prediction the linear model, which is conceptually not a good idea whereas MPC-NPSL algorithm uses the nonlinear Wiener–Hammerstein model of the process, but in a simplified way, i.e. the transfer functions of the linear dynamic parts of the model are multiplied by the gain of the nonlinear steady-state block for the current operating point, defined by the value of the variable  $v(k)$  [Eq. (53)]. Figure 6 depicts



**Fig. 5** Simulation results of the linear MPC algorithm;  $\lambda = 8000$



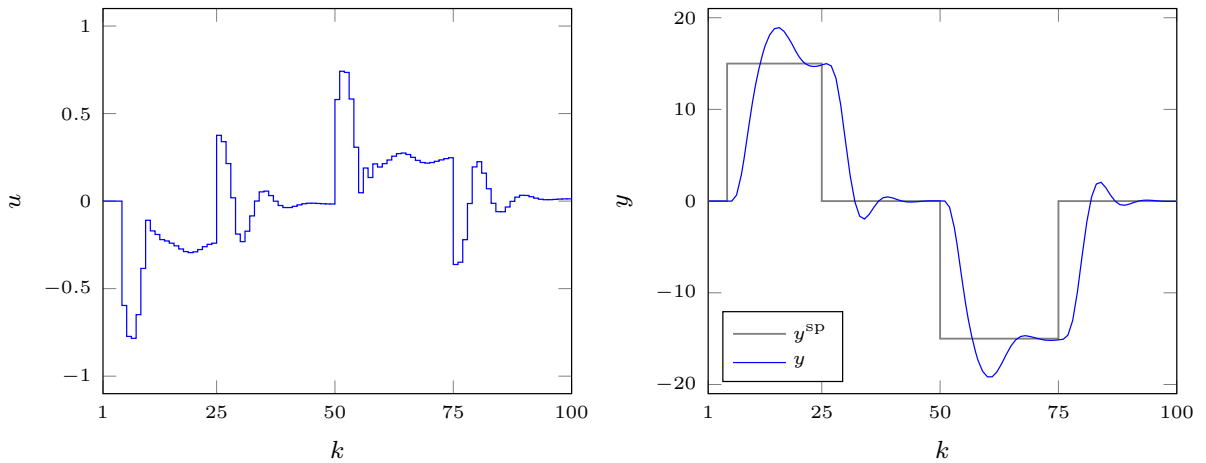
**Fig. 6** Simulation results of the MPC-NPSL algorithm with simplified on-line model linearisation and quadratic optimisation;  $\lambda = 1$

the obtained simulation results when the penalty factor  $\lambda$  has its default value 1. Unfortunately, although the obtained control quality is much better than in the case of the linear MPC algorithm, i.e. the output trajectory of the process is stable and there is no steady-state error, the MPC-NPSL algorithm gives big overshoot. It is because the changes of the set-point trajectory are fast and significant. It may be also noted that the input trajectory is characterised by big changes, even from the minimal to the maximal value of the manipulated variable. Similarly to the linear MPC algorithm, it is also possible to improve the MPC-NPSL algorithm by increasing the penalty  $\lambda$ . Figure 7 shows the trajectories obtained for  $\lambda = 250$ . Because the MPC-NPSL algorithm uses more accurate approximation of the system

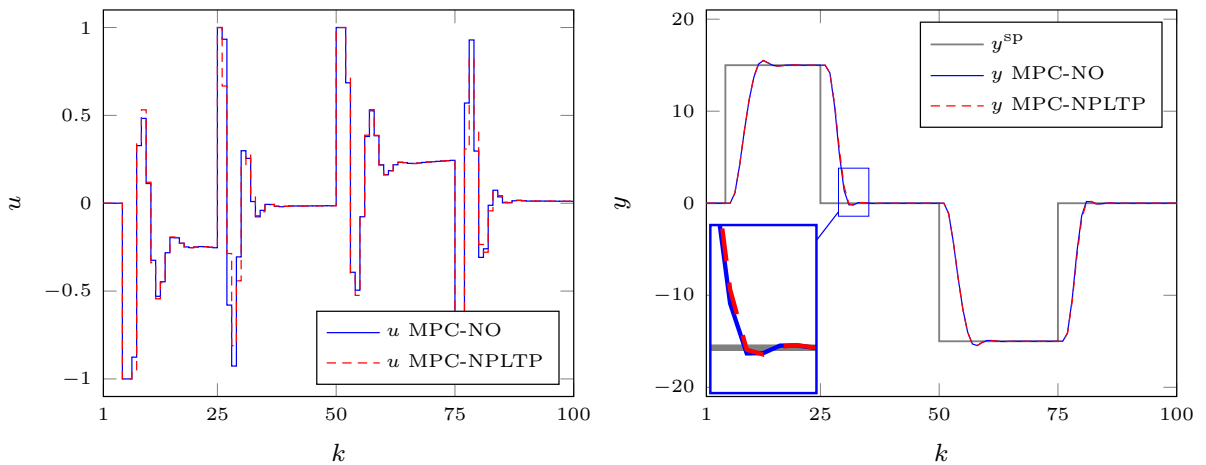
than the linear MPC approach does, it requires much a lower value of  $\lambda$  (250) in comparison with that necessary in the linear MPC algorithm (8000). In consequence, the MPC-NPSL (with increased  $\lambda$ ) is faster than the linear MPC strategy, but it is still quite slow and its overshoot is unacceptable.

Next, the MPC-NPLPT algorithm with on-line trajectory linearisation is considered. When compared with the MPC-NPSL strategy, it uses conceptually a more advanced approximation method of the predicted output trajectory. It is because the MPC-NPLPT scheme directly finds a linear approximation of the predicted output trajectory over the prediction horizon, linearisation is carried out for a future input trajectory defined over the control horizon, whereas in the





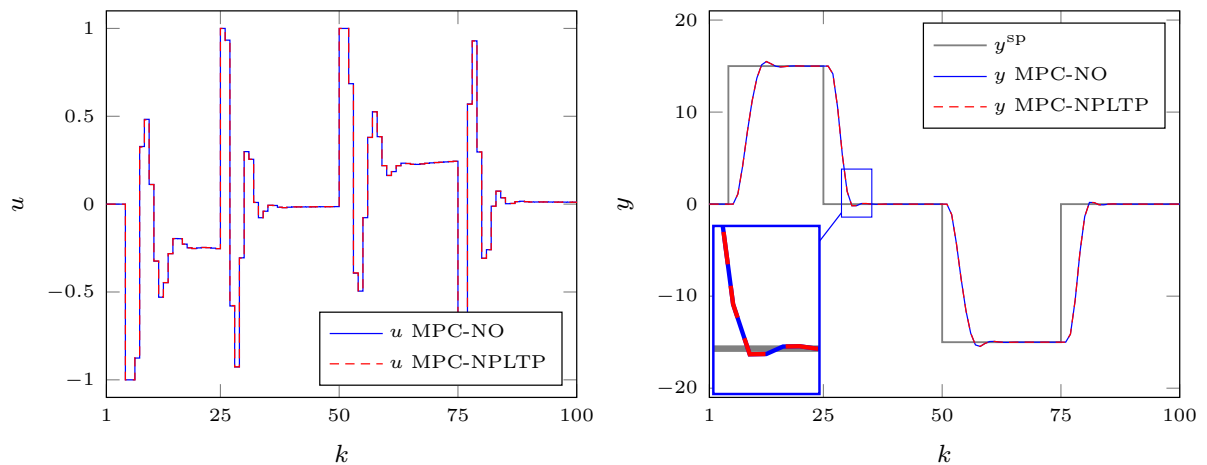
**Fig. 7** Simulation results of the MPC-NPSL algorithm with simplified on-line model linearisation and quadratic optimisation;  $\lambda = 250$



**Fig. 8** Simulation results: the MPC-NPLPT algorithm with on-line trajectory linearisation and quadratic optimisation with  $t_{max} = 1$  (solid line), the MPC-NO algorithm with on-line nonlinear optimisation (dashed line); in both algorithms  $\lambda = 1$

MPC-NPSL algorithm only a linear approximation of the model is found for the current operating point of the process and the forced trajectory  $G(k)\Delta u(k)$  in Eq. (25) is calculated from the same linearised model applied recurrently over the prediction horizon. Therefore, the MPC-NPLPT algorithm is likely to give more accurate approximation of the predicted output trajectory than the MPC-NPSL one and to result in better control quality. Figure 8 compares the trajectories of the MPC-NPLPT algorithm with those obtained in the MPC-NO in which the full nonlinear Wiener-Hammerstein model is used for prediction and optimisation of the future control policy, and no approximation of the model or the predicted output trajec-

tory is used. In both algorithms the parameter  $\lambda$  has its default value 1. Nevertheless, in contrast to the linear MPC algorithm and the NPC-NPSL one (Figs. 4, 6), the MPC-NPLPT algorithm gives very good, precise, steady-state error-free control, and overshoot is very small. Furthermore, it may be noticed that the trajectories of the MPC-NPLPT algorithm are very similar to those obtained in the truly nonlinear MPC-NO strategy. It is interesting to point out that very good control quality of the MPC-NPLPT algorithm is obtained even though only one internal iteration is performed at each sampling instant ( $t_{max} = 1$ ), which means that the predicted output trajectory is linearised only once at each instant. In order to increase accuracy of the



**Fig. 9** Simulation results: the MPC-NPLPT algorithm with on-line trajectory linearisation and quadratic optimisation with  $t_{max} = 5$ ,  $\delta_u = \delta_y = 1$  (solid line), the MPC-NO algorithm with on-line nonlinear optimisation (dashed line); in both algorithms  $\lambda = 1$

MPC-NPLPT algorithm, the maximal possible number of internal iterations is increased to 5, and the additional parameters are  $\delta_u = \delta_y = 1$ . The obtained simulation results are shown in Fig. 9. In this case the trajectories of the MPC-NPLPT algorithm with on-line trajectory linearisation and quadratic optimisation are practically the same as those of the MPC-NO algorithm.

In order to conveniently compare effectiveness of the discussed MPC algorithms, two indices are defined. The first one is the sum of squared errors (SSE)

$$SSE = \sum_{k=1}^{100} (y^{SP}(k) - y(k))^2$$

It is interesting to point out that the SSE index is not actually minimised in MPC. That is why the second, relative index takes into account the discrepancy between any MPC algorithm and theoretically the best MPC-NO strategy

$$D = \sum_{k=1}^{100} (y_{MPC-NO}(k) - y(k))^2 \tag{54}$$

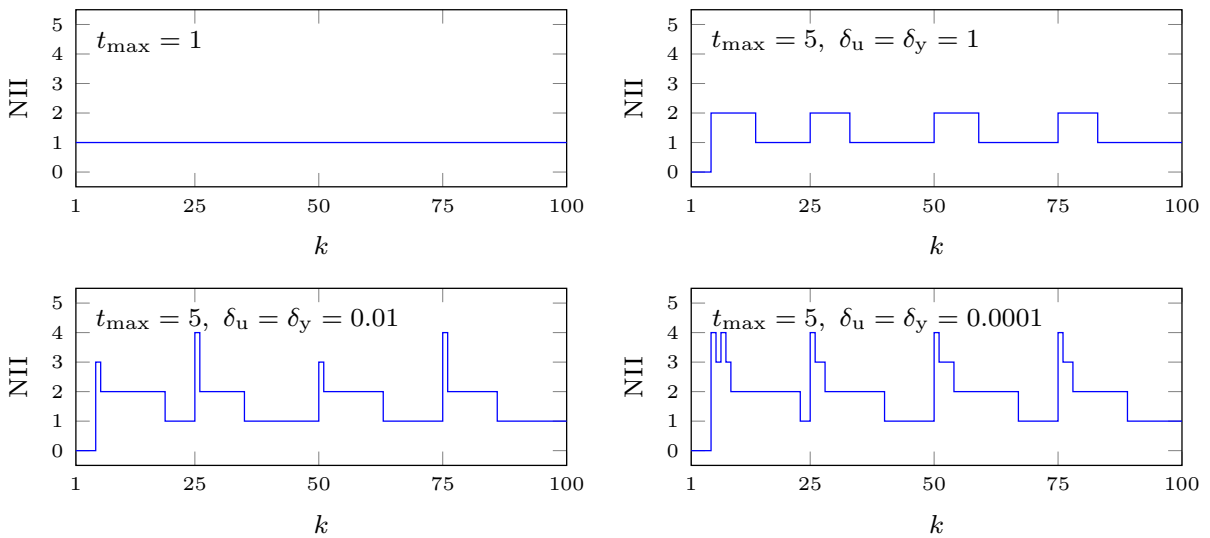
where  $y_{MPC-NO}(k)$  denotes the output of the process when it is controlled by the MPC-NO algorithm, and  $y(k)$  is the output if a different MPC algorithm is used. Table 1 compares all evaluated MPC algorithms (linear MPC, MPC-NPSL, MPC-NPLPT) in terms of the indices SSE and  $D$ , as well as computational burden (MFLOPS). The linear MPC strategy is very bad, the

MPC-NPSL is much better, but only the MPC-NPLPT strategy makes it possible to obtain control quality comparable with that typical of the MPC-NO strategy. As many as four versions of the MPC-NPLPT algorithms are compared: with one internal iteration and three versions with maximally 5 iterations, for different values of the tuning parameters  $\delta_u$  [(Eq. (49))] and  $\delta_y$  [(Eq. (48))]. Additionally, the total number of internal iterations is given in Table 1 for the MPC-NPLPT algorithm. In general, the smaller the values of  $\delta_u$  and  $\delta_y$ , the smaller the discrepancy between the MPC-NPLPT algorithm and the MPC-NO one, and, at the same time, more internal iterations are necessary, which increases computational burden. The MPLPT algorithm with only one internal iteration is as many as 15.84 times more computationally efficient than the MPC-NO one. When 5 internal iterations are allowed, for  $\delta_u = \delta_y = 1$  this factor drops to 11, when  $\delta_u = \delta_y = 10^{-2}$ , to 9.37 and when  $\delta_u = \delta_y = 10^{-4}$ , to 7.69. Figure 10 shows the number of internal iterations in the consecutive sampling instants of four compared versions of the MPC-NPLPT algorithm. It is interesting to note that when  $\delta_u = \delta_y = 1$ , which results in very good control quality, in the majority of sampling instants only one internal iteration is sufficient. More internal iterations are necessary when the set point changes.

In many practical control systems not only the range of the manipulated variable, but also its speed of change is constrained. It is possible to slower the MPC algorithms by increasing the penalty factor  $\lambda$ . Figures 11 and 12 depict simulation results of the MPC-NPLPT

**Table 1** Control accuracy (SSE), the relative accuracy criterion ( $D$ ) and computational burden (MFLOPS) of compared MPC algorithms, for the MPC-NPLPT strategy the sum of internal iterations (NII) is also given; in all algorithms  $\lambda = 1$

Algorithm	SSE	$D$	MFLOPS	NII
Linear MPC	$8.9758 \times 10^3$	$5.6648 \times 10^3$	8.79	–
MPC-NPSL	$3.3507 \times 10^3$	$2.5582 \times 10^2$	9.39	–
MPC-NPLPT, $t_{\max} = 1$	$3.2471 \times 10^3$	$1.0681 \times 10^0$	17.04	100
MPC-NPLPT, $t_{\max} = 5, \delta_u = \delta_y = 10^0$	$3.2226 \times 10^3$	$1.4549 \times 10^{-4}$	24.55	130
MPC-NPLPT, $t_{\max} = 5, \delta_u = \delta_y = 10^{-2}$	$3.2225 \times 10^3$	$1.0095 \times 10^{-4}$	28.81	150
MPC-NPLPT, $t_{\max} = 5, \delta_u = \delta_y = 10^{-4}$	$3.2224 \times 10^3$	$3.9883 \times 10^{-7}$	35.10	179
MPC-NO	$3.2224 \times 10^3$	0	270.01	–

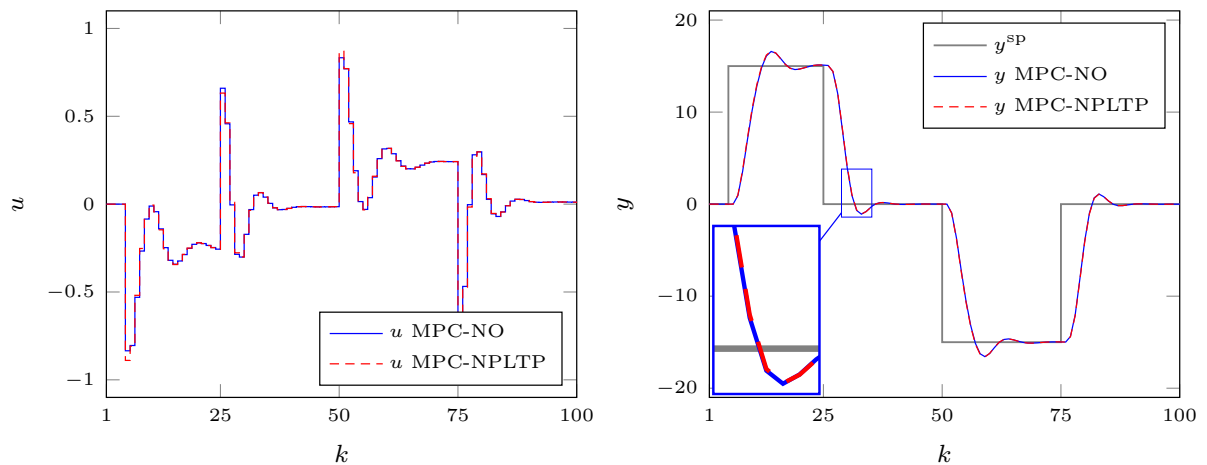


**Fig. 10** The number of internal iterations (NII) in the consecutive sampling instants of four versions of the MPC-NPLPT algorithm with on-line trajectory linearisation and quadratic optimisation;  $\lambda = 1$

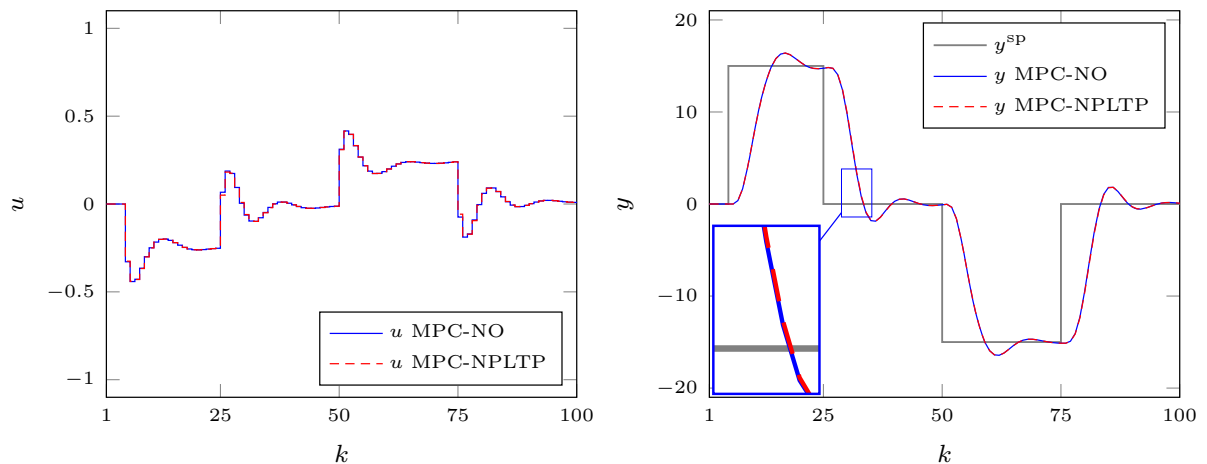
and MPC-NO algorithms for  $\lambda = 100$  and  $\lambda = 1000$ , respectively. The bigger the penalty parameter, the slower the input and output trajectories of the process and the bigger the overshoot. Unfortunately, it is impossible to guarantee that all the increments of the manipulated variable for each sampling instant fulfil any fixed constraint. That is why the rate constraint is taken into account in the optimisation problems. The trajectories obtained in the MPC-NPLPT and MPC-NO algorithms with the additional constraint  $\Delta u^{\max} = 0.25$  are depicted in Fig. 13. The additional constraint is satisfied at each sampling instant. Also in this case there are no noticeable differences between the MPC-NPLPT and MPC-NO compared algorithms, although during the presented simulations the parameters are  $\delta_u = \delta_y = 1, t_{\max} = 5$ . Table 2 compares all discussed algorithms in

terms of the indices SSE and  $D$ , computational burden (MFLOPS), and in the case of the MPC-NPLPT algorithm the number of internal iterations is given. The additional constraint increases computational burden of all MPC algorithms (when compared with Table 1), but still the MPC-NPLPT algorithm with trajectory linearisation and quadratic optimisation is many times more computationally efficient than the MPC-NO one.

An interesting issue is to verify effectiveness of the MPC algorithms when the process is affected by unmeasured output noise (with normal distribution with zero mean and standard deviation 0.1). The obtained simulation results of the MPC-NPLPT and MPC-NO algorithms are depicted in Fig. 14, and Table 3 compares all discussed algorithms in terms of the indices SSE and  $D$ , computational burden



**Fig. 11** Simulation results: the MPC-NPLPT algorithm with on-line trajectory linearisation and quadratic optimisation with  $t_{\max} = 5$ ,  $\delta_u = \delta_y = 1$  (solid line), the MPC-NO algorithm with on-line nonlinear optimisation (dashed line); in both algorithms  $\lambda = 100$

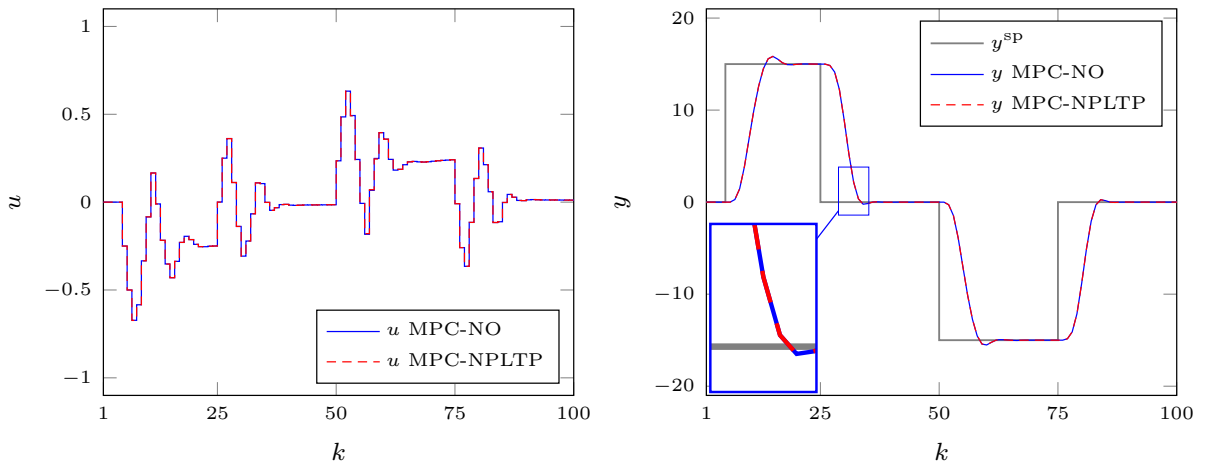


**Fig. 12** Simulation results: the MPC-NPLPT algorithm with on-line trajectory linearisation and quadratic optimisation with  $t_{\max} = 5$ ,  $\delta_u = \delta_y = 1$  (solid line), the MPC-NO algorithm with on-line nonlinear optimisation (dashed line); in both algorithms  $\lambda = 1000$

(MFLOPS) and the number of internal iterations (for the MPC-NPLPT algorithm). Finally, Fig. 15 compares the MPC-NPLPT and MPC-NO algorithms when the additional constraint  $\Delta u^{\max} = 0.25$  is taken into account and the process is affected by unmeasured noise, and the values of all criteria SSE,  $D$ , MFLOPS and NII are given in Table 4. It is necessary to point out that in all cases the MPC-NPLPT algorithm with trajectory linearisation and quadratic optimisation gives the trajectories practically the same as the MPC-NO approach with nonlinear on-line linearisation, whereas its computational burden is much lower.

## 6 Conclusions

This work details two general nonlinear MPC algorithms for dynamic processes described by cascade Wiener–Hammerstein models, which consist of a nonlinear steady-state block sandwiched by two linear dynamic ones. The algorithms are computationally efficient, and since only quadratic optimisation problems are solved on-line, no nonlinear optimisation is necessary. In both cases the inverse steady-state block is not necessary, which is a very frequent approach to control based on cascade models.



**Fig. 13** Simulation results: the MPC-NPLTP algorithm with on-line trajectory linearisation and quadratic optimisation with  $t_{\max} = 5$ ,  $\delta_u = \delta_y = 1$  (solid line), the MPC-NO algorithm with on-line nonlinear optimisation (dashed line); in both algorithms

$\lambda = 1$  and the additional constraint  $\Delta u^{\max} = 0.25$  imposed on the rate of change of the manipulated variable is taken into account

**Table 2** Control accuracy (SSE), the relative accuracy criterion ( $D$ ) and computational burden (MFLOPS) of compared MPC algorithms, for the MPC-NPLPT strategy the sum of internal

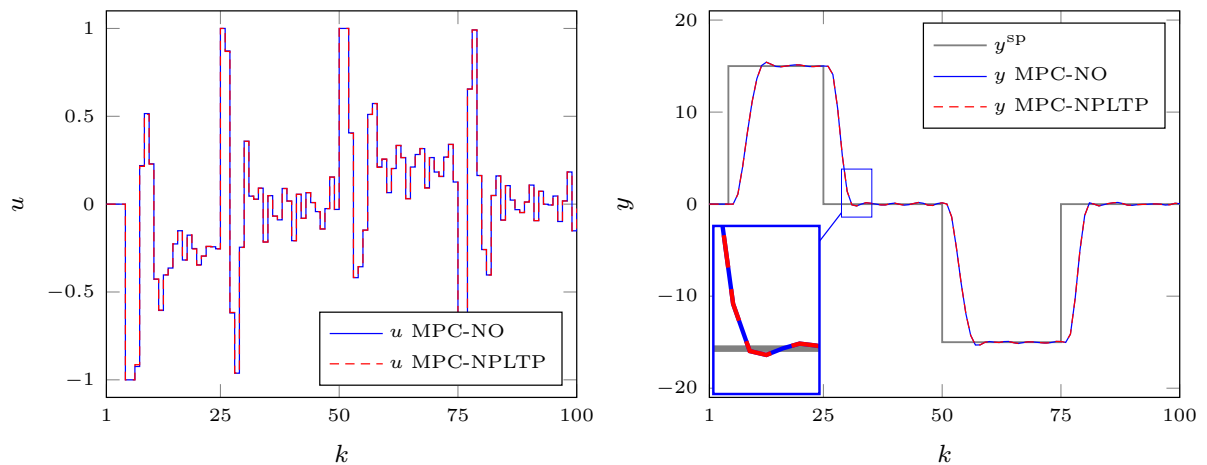
iterations (NII) is also given; in all algorithms  $\lambda = 1$  and the additional constraint  $\Delta u^{\max} = 0.25$  imposed on the rate of change of the manipulated variable is taken into account

Algorithm	SSE	$D$	MFLOPS	NII
Linear MPC	$6.4735 \times 10^3$	$3.5184 \times 10^3$	13.61	–
MPC-NPSL	$4.5174 \times 10^3$	$1.6602 \times 10^2$	13.16	–
MPC-NPLPT, $t_{\max} = 1$	$4.3867 \times 10^3$	$2.2540 \times 10^{-1}$	20.64	100
MPC-NPLPT, $t_{\max} = 5$ , $\delta_u = \delta_y = 10^0$	$4.3949 \times 10^3$	$2.8476 \times 10^{-4}$	32.86	139
MPC-NPLPT, $t_{\max} = 5$ , $\delta_u = \delta_y = 10^{-2}$	$4.3949 \times 10^3$	$2.8108 \times 10^{-4}$	37.78	158
MPC-NPLPT, $t_{\max} = 5$ , $\delta_u = \delta_y = 10^{-4}$	$4.3948 \times 10^3$	$7.0495 \times 10^{-6}$	43.76	180
MPC-NO	$4.3948 \times 10^3$	0	336.57	–

**Table 3** Control accuracy (SSE), the relative accuracy criterion ( $D$ ) and computational burden (MFLOPS) of compared MPC algorithms, for the MPC-NPLPT strategy the sum of internal

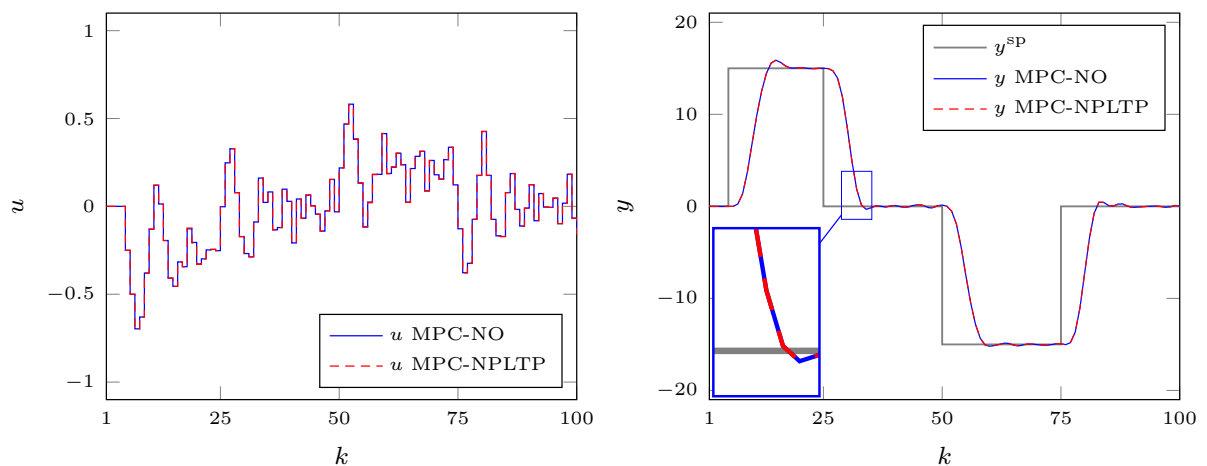
iterations (NII) is also given; in all algorithms  $\lambda = 1$  and the process is affected by unmeasured noise

Algorithm	SSE	$D$	MFLOPS	NII
Linear MPC	$8.4341 \times 10^3$	$5.5010 \times 10^3$	8.68	–
MPC-NPSL	$3.3463 \times 10^3$	$1.9925 \times 10^2$	9.41	–
MPC-NPLPT, $t_{\max} = 1$	$3.2132 \times 10^3$	$9.2140 \times 10^{-1}$	17.05	100
MPC-NPLPT, $t_{\max} = 5$ , $\delta_u = \delta_y = 10^0$	$3.1935 \times 10^3$	$3.9639 \times 10^{-4}$	24.40	129
MPC-NPLPT, $t_{\max} = 5$ , $\delta_u = \delta_y = 10^{-2}$	$3.1935 \times 10^3$	$3.9639 \times 10^{-4}$	34.02	186
MPC-NPLPT, $t_{\max} = 5$ , $\delta_u = \delta_y = 10^{-4}$	$3.1935 \times 10^3$	$3.9639 \times 10^{-7}$	41.07	223
MPC-NO	$3.1934 \times 10^3$	0	266.42	–



**Fig. 14** Simulation results: the MPC-NPLTP algorithm with on-line trajectory linearisation and quadratic optimisation with  $t_{\max} = 5$ ,  $\delta_u = \delta_y = 1$  (solid line), the MPC-NO algorithm with

on-line nonlinear optimisation (dashed line); in both algorithms  $\lambda = 1$  and the process is affected by unmeasured noise



**Fig. 15** Simulation results: the MPC-NPLTP algorithm with on-line trajectory linearisation and quadratic optimisation with  $t_{\max} = 5$ ,  $\delta_u = \delta_y = 1$  (solid line), the MPC-NO algorithm with on-line nonlinear optimisation (dashed line); in both algorithms

$\lambda = 1$ , the additional constraint  $\Delta u^{\max} = 0.25$  imposed on the rate of change of the manipulated variable is taken into account and the process is affected by unmeasured noise

The first algorithm (MPC-NPSL) is conceptually quite simple: at each sampling instant on-line the gain of the nonlinear block is found and used to update the parameters of the linear model. Such a linear approximation of the model is used to calculate the predicted output trajectory and the future control increments, and the influence of the past is determined from the nonlinear Wiener–Hammerstein model. A clear disadvantage of that approach is the fact that the model is linearised

on-line in a simplified way and the same linearised model is used for the prediction over the whole prediction horizon. That is why the MPC algorithm with simplified model linearisation, although for the considered heat exchanger benchmark it works much better than the rudimentary MPC algorithms based on a linear model, gives big overshoot when the set point changes fast and significantly.



**Table 4** Control accuracy (SSE), the relative accuracy criterion ( $D$ ) and computational burden (MFLOPS) of compared MPC algorithms, for the MPC-NPLPT strategy the sum of internal iterations (NII) is also given; in all algorithms  $\lambda = 1$ , the addi-

tional constraint  $\Delta u^{\max} = 0.25$  imposed on the rate of change of the manipulated variable is taken into account and the process is affected by unmeasured noise

Algorithm	SSE	$D$	MFLOPS	NII
Linear MPC	$1.5876 \times 10^3$	$1.0167 \times 10^4$	13.80	–
MPC-NPSL	$4.3896 \times 10^3$	$1.6696 \times 10^2$	13.23	–
MPC-NPLPT, $t_{\max} = 1$	$4.2469 \times 10^3$	$3.0850 \times 10^{-1}$	20.52	100
MPC-NPLPT, $t_{\max} = 5$ , $\delta_u = \delta_y = 10^0$	$4.2526 \times 10^3$	$4.0882 \times 10^{-3}$	32.74	138
MPC-NPLPT, $t_{\max} = 5$ , $\delta_u = \delta_y = 10^{-2}$	$4.2538 \times 10^3$	$4.3179 \times 10^{-4}$	40.97	184
MPC-NPLPT, $t_{\max} = 5$ , $\delta_u = \delta_y = 10^{-4}$	$4.2538 \times 10^3$	$3.3998 \times 10^{-5}$	48.42	206
MPC-NO	$4.2539 \times 10^3$	0	330.16	–

The second strategy (MPC-NPLPT) uses a more advanced linearisation method. At each sampling instant on-line a linear approximation of the predicted output trajectory is found over the whole prediction horizon. It is a conceptually better approach in comparison with the first one, in which the model itself is only linearised for the current operating point. Indeed, for the considered heat exchanger system the MPC algorithm with trajectory linearisation gives the same trajectory as the MPC-NO algorithm with nonlinear optimisation repeated at each sampling instant on-line. Furthermore, it is computationally efficient in two respects: qualitatively and quantitatively. Firstly, it needs only quadratic optimisation. Secondly, when compared with the MPC-NO algorithm it is many times less computationally demanding.

In contrast to the extended dynamic control (EDMC) algorithm [36] which extends the linear DMC algorithm by using a time-varying disturbance vector which captures the effect of nonlinearities, in the described MPC strategies the effect of the past is calculated directly from the full nonlinear model, the horizon of dynamics is not used (only prediction and control horizons are necessary), and it is not required that the process is stable and it does not contain integration.

A natural direction of future research is the incorporation of the Wiener–Hammerstein model in economic MPC [37, 38] or in set-point optimisation cooperating with MPC [1, 12]. The mentioned approaches tightly integrate economic optimisation and feedback MPC.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## References

1. Tatjewski, P.: *Advanced Control of Industrial Processes. Structures and Algorithms*. Springer, London (2007)
2. Qin, S.J., Badgwell, T.A.: A survey of industrial model predictive control technology. *Control Eng. Pract.* **11**, 733–764 (2003)
3. Dubay, R., Hassan, M., Li, C., Charest, M.: Finite element based model predictive control for active vibration suppression of a one-link flexible manipulator. *ISA Trans.* **53**, 1609–1619 (2014)
4. Eski, İ., Temürleik, A.: Design of neural network-based control systems for active steering system. *Nonlinear Dyn.* **73**, 1443–1454 (2014)
5. Preglej, A., Rehrl, J., Schwingshackl, D., Steiner, I., Horn, M., Škrjanc, I.: Energy-efficient fuzzy model-based multi-variable predictive control of a HVAC system. *Energ. Build.* **82**, 520–533 (2014)
6. Sardarmehni, T., Rahmani, H., Menhaj, M.B.: Robust control of wheel slip in anti-lock brake system of automobiles. *Nonlinear Dyn.* **76**, 125–138 (2014)
7. Schnelle, P.D., Rollins, D.L.: Industrial model predictive control technology as applied to continuous polymerization processes. *ISA Trans.* **36**, 281–292 (1998)
8. Sudiby, S., Murat, M.N., Aziz, N.: Simulated annealing-Particle Swarm Optimization (SA-PSO): particle distribution study and application in Neural Wiener-based NMPC. In: *The 10th Asian Control Conference* (2015)
9. Wu, Z., Xia, X., Zhu, B.: Model predictive control for improving operational efficiency of overhead cranes. *Nonlinear Dyn.* **79**, 2639–2657 (2015)

10. Bigdeli, N., Haeri, M.: Predictive functional control for active queue management in congested TCP/IP networks. *ISA Trans.* **48**, 107–121 (2009)
11. Grosso, J.M., Ocampo-Martínez, C., Puig, V.: Learning-based tuning of supervisory model predictive control for drinking water networks. *Eng. Appl. Artif. Intel.* **26**, 1741–1750 (2013)
12. Ławryńczuk, M.: Computationally Efficient Model Predictive Control Algorithms: A Neural Network Approach. *Studies in Systems, Decision and Control*, vol. 3, Springer, Cham (2014)
13. Giri, F., Bai E.W. (Eds.): *Block-oriented Nonlinear System Identification. Lecture Notes in Control and Information Sciences*, vol. **404**, Springer, Berlin (2010)
14. Janczak, A.: Identification of Nonlinear Systems Using Neural Networks and Polynomial Models. A Block-Oriented Approach. *Lecture Notes in Control and Information Sciences*, vol. **310**, Springer, Berlin (2004)
15. Chan, K.H., Bao, J.: Model predictive control of Hammerstein systems with multivariable nonlinearities. *Ind. Eng. Chem. Res.* **46**, 168–180 (2007)
16. Fruzzetti, K.P., Palazoğlu, A., McDonald, K.A.: Nonlinear model predictive control using Hammerstein models. *J. Proc. Control* **7**, 31–41 (1997)
17. Harnischmacher, G., Marquardt, W.: Nonlinear model predictive control of multivariable processes using block-structured models. *Control Eng. Pract.* **15**, 1238–1256 (2007)
18. Ławryńczuk, M.: Suboptimal nonlinear predictive control based on multivariable neural Hammerstein models. *Appl. Intell.* **32**, 173–192 (2010)
19. Cervantes, A.L., Agamnoni, O.E., Figueroa, J.L.: A nonlinear model predictive control system based on Wiener piecewise linear models. *J. Proc. Control* **13**, 665–666 (2003)
20. Ławryńczuk, M.: Computationally efficient nonlinear predictive control based on neural Wiener models. *Neurocomputing* **74**, 401–417 (2010)
21. Peng, J., Dubay, R., Hernandez, J.M., Abu-Ayyad, M.: A Wiener neural network-based identification and adaptive generalized predictive control for nonlinear SISO systems. *Ind. Eng. Chem. Res.* **50**, 7388–7397 (2011)
22. Al Seyab, R.K., Cao, Y.: Nonlinear model predictive control for the ALSTOM gasifier. *J. Proc. Control* **16**, 795–808 (2006)
23. Ławryńczuk, M.: Practical nonlinear predictive control algorithms for neural Wiener models. *J. Proc. Control* **23**, 696–714 (2013)
24. Ding, B., Ping, X.: Dynamic output feedback model predictive control for nonlinear systems represented by Hammerstein–Wiener model. *J. Proc. Control* **22**, 1773–1784 (2012)
25. Patikirikoral, T., Wang, L., Colman, A., Han, J.: Hammerstein–Wiener nonlinear model based predictive control for relative QoS performance and resource management of software systems. *Control. Eng. Pract.* **20**, 49–61 (2012)
26. Ławryńczuk, M.: Nonlinear predictive control for Hammerstein–Wiener systems. *ISA Trans.* **55**, 49–62 (2015)
27. Bloemen, H.H.J., Boom, T.J.J., Verbruggen, H.B.: Model-based predictive control for Hammerstein–Wiener systems. *Int. J. Contr.* **74**, 482–495 (2001)
28. Benyó, I., Kovács, J., Mononen, J., Kortela, U.: Modelling of steam temperature dynamics of a superheater. *Int. J. Simul.* **6**, 3–9 (2005)
29. Crama, P., Rolain, Y.: Broadband measurement and identification of a Wiener–Hammerstein model for an RF amplifier. In: *ARFTG Conference Digest*, pp. 49–57 (2002)
30. Haryanto, A., Hong, K.-S.: Maximum likelihood identification of Wiener–Hammerstein models. *Mech. Systems Signal Proces.* **41**, 54–70 (2013)
31. Oliver, J.A., Prieto, R., Cobos, J.A., Garcia, O., Alou, P.: Hybrid Wiener–Hammerstein structure for grey-box modeling of DC–DC converters. In: *The 24th Annual IEEE Conference Applied Power Electronics Conference and Exposition* (2009)
32. Pan, J., Cheng, C.-H.: Wiener–Hammerstein model based electrical equalizer for optical communication systems. *J. Lightwave Technol.* **29**, 2454–2459 (2011)
33. Piroddi, L., Farina, M., Lovera, M.: Black box model identification of nonlinear input-output models: a Wiener–Hammerstein benchmark. *Contr. Eng. Pract.* **20**, 1109–1118 (2012)
34. Deschênes, J.-S., St-Onge P.N.: Achievable performances for basic perturbation-based extremum seeking control in Wiener–Hammerstein plants. In: *The 52nd IEEE Conference on Decision and Control*, pp. 2991–2998 (2013)
35. Nocedal, J., Wright, S.J.: *Numerical Optimization*. Springer, Berlin (2006)
36. Peterson, T., Hernandez, E., Arkun, Y., Schork, F.J.: A nonlinear DMC algorithm and its application to a semibatch polymerization reactor. *Chem. Eng. Sci.* **47**, 737–753 (1992)
37. Alanqar, A., Durand, H., Christofides, P.D.: On identification of well-conditioned nonlinear systems: application to economic model predictive control of nonlinear processes. *AIChE J.* **61**, 3353–3373 (2015)
38. Alanqar, A., Ellis, M., Christofides, P.D.: Economic model predictive control of nonlinear process systems using empirical models. *AIChE J.* **61**, 816–830 (2015)