Nonlinear Dyn (2018) 92:1001–1021
https://doi.org/10.1007/s11071-018-4105-5

ORIGINAL PAPER

Nonlinear model predictive control based on piecewise linear Hammerstein models

Jian Zhang · Kwai-Sang Chin · Maciej Ławryńczuk

Received: 16 April 2017 / Accepted: 26 January 2018 / Published online: 15 February 2018

Abstract This paper develops a nonlinear model predictive control (MPC) algorithm for dynamic systems represented by piecewise linear (PWL) Hammerstein models. At each sampling instant, the predicted output trajectory is linearized online at an assumed input trajectory such that the control actions can be easily calculated by solving a quadratic programming optimization problem, and such linearization and optimization may be repeated a few times for good linear approximation accuracy. A three-step procedure is developed to linearize a PWL function, where the derivatives of a PWL function are obtained by a computationally efficient look-up table approach. Unlike many existing MPC algorithms for Hammerstein systems, it does not require the inversion of static nonlinearity and can directly cope with input constraints even in multivariable systems. Two benchmark chemical reactors are studied to illustrate the effectiveness of the proposed algorithm.

J. Zhang (✉) · K.-S. Chin
Department of System Engineering and Engineering Management, City University of Hong Kong, Kowloon Tong, Hong Kong
e-mail: jzhang398-c@my.cityu.edu.hk

K.-.S Chin
e-mail: mekschin@cityu.edu.hk

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

Keywords Nonlinear model predictive control · Hammerstein model · Piecewise linear function · Multistep linearization · Chemical reactor

1 Introduction

Model predictive control (MPC) refers to a class of control algorithms that employ an explicit dynamic model of a controlled process to predict its future outputs, and determine the control actions through optimization [1]. MPC has been one of the most successful control techniques for the industrial processes with multivariable coupling, constraints, and time-delay [2,3]. With linear MPC becoming mature, nonlinear MPC becomes the focus of current research as it can contribute to good control performance for highly nonlinear processes.

Many nonlinear models have been integrated in MPC algorithms, such as first-principle models [4], neural networks [5], support vector machines (SVM) [6], and block-oriented models [7–10]. In particular, a Hammerstein model, which consists of a static nonlinear model followed by a linear dynamic model, has been demonstrated to be able to adequately approximate a large number of processes, such as a binary distillation column [7], a pH neutralization process [11], a continuous stirred tank reactor [12], a solid oxide fuel cell [13], a stretch reflex process [14], a turntable servo system [15], and so on. Compared with other nonlinear models, the Hammerstein model has the advantages of simple structure and easy identification [7,8], and thus is considered here.

In the most typical case, a polynomial is used as the static part of the Hammerstein model. A significant drawback of this model is that, for accurately approximating complex nonlinearities, the polynomial order is likely to be very high [16]. This will cause the difficulty of identification and oscillatory interpolation. In the view of nonlinear approximation, neural networks are excellent approximators [5]. Unfortunately, training a neural network is not easy. Nonlinear optimization problems have to be solved to compute the network model parameters, and a number of networks should be trained and compared to get a good model. Support vector machines (SVM) are also employed to act the static part of a Hammerstein model [17], but it has the disadvantage of lacking spareness, i.e., including a large number of support vectors and corresponding parameters. For these reasons, it is necessary to find a suitable nonlinear approximator for Hammerstein models. A piecewise linear (PWL) function is an interesting alternative. Based on the so-called simplicial partition method, the canonical PWL representation is studied in [18,19]. The parameters of canonical PWL functions can be identified by solving a least-square problem; they can be easily calculated by using the existing PWL toolbox [20]. It has been proven that PWL functions can uniformly approximate any Lipschitz-continuous function defined on a compact domain. Moreover, PWL functions do not suffer from the problems of oscillatory interpolation and lacking spareness.

Several MPC algorithms have been developed for dynamic systems represented by Hammerstein models. The main focus of these algorithms is how to deal with the static nonlinearity for optimizing control laws. A pioneering study can be found in [21], where the Hammerstein model is directly integrated into the MPC controller and the corresponding control law is calculated by solving a complex non-convex nonlinear optimization problem. By transforming the nonlinearity to a polytopic description, the Robust MPC (RMPC) has been designed for Hammerstein models [22,23]. This algorithm just requires to solve a quadratic programming (QP) problem with linear matrix inequality (LMI) constraints, but the conservativeness and the online computational burden should be carefully considered. By treating input nonlinearities as unknown disturbances, the linear MPC with an input disturbance model is also studied for Hammerstein systems with unknown input nonlinearities [24]. The most common and the most used control algorithms for Hammerstein models are the nonlinearity inversion-based MPC algorithms which consist of a linear controller followed by the inversion of input nonlinearity [7,9,10]. For multivariable systems with complex nonlinearities, such algorithms have to calculate numerical inversion of the nonlinearities online and require extra transformation of input constraints. An another idea is to find a linear approximation of the model or of the predicted process trajectory which makes formulation of a QP MPC problem possible [5,17,25].

This paper develops a nonlinear MPC algorithm based on PWL Hammerstein models. In contrast to the aforementioned MPC algorithms, the proposed MPC algorithm does not require inversion of input nonlinear block, and thus, it can directly integrate input constraints without any transformation. To reduce computational burden, at each sampling period, the predicted output trajectory is firstly linearized at an assumed input trajectory, and then the optimal control actions are simply calculated by solving a QP problem. In particular, due to the character that a PWL function becomes a linear function at a specific input subregion, the derivatives used in the linearization process are obtained in a computationally efficient look-up table style. The developed control algorithm can integrate various disturbance models, which makes it possible to achieve better control performance than the controllers just based on the traditional output disturbance model. This paper extends the work of [17,25], where only output disturbance models are considered, SVM and neural networks are, respectively, used to act the nonlinear part of the model, and the derivatives of static nonlinearities are calculated online. Two benchmark systems, a continuous stirred tank reactor (CSTR) and a PH neutralization reactor, are employed to show the advantages of the proposed algorithm. The simulation results illustrate that the algorithm can make a good balance between control performance and computational efficiency. Namely, it can give almost same control accuracy with that of the MPC algorithm with nonlinear optimization, at the cost of slightly higher computational burden than the linear MPC.

The rest of this paper is organized as follows. Section 2 introduces the PWL Hammerstein model and the canonical representation for PWL functions. Section 3 details development of the nonlinear MPC based on the PWL Hammerstein model. Section 4 gives the simulation results for the two benchmark systems. Section 5 concludes the paper.

2 Preliminaries

2.1 Piecewise linear Hammerstein model

The controlled dynamic system is represented by the following discrete-time, nonlinear, state-space model

$$\begin{cases} \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ \boldsymbol{y}_k = g(\boldsymbol{x}_k) \end{cases} \tag{1}$$

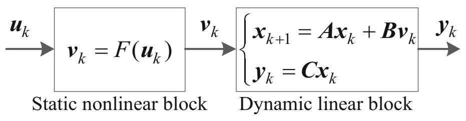
where $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ is the system input vector (the vector of manipulated variables), $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ is the system state vector, $\boldsymbol{y}_k \in \mathbb{R}^{n_y}$ is the system output vector (the vector of controlled variables), k is the sample time, $f : \mathbb{R}^{n_u+n_x} \rightarrow \mathbb{R}^{n_x}$ and $g : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y}$ are smooth nonlinear functions.

A Hammerstein model is employed to approximate the above nonlinear process. As shown in Fig. 1, the Hammerstein model consists of a static nonlinear block followed by a dynamic linear block. It can be described by the following equations

$$\begin{cases} \boldsymbol{v}_k = F(\boldsymbol{u}_k) \\ \boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{v}_k \\ \boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k \end{cases} \tag{2}$$

where $\boldsymbol{v}_k \in \mathbb{R}^{n_v}$ is an intermediate variable vector between model blocks, $F : \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_v}$ is a static nonlinear function. The model matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are of dimensionality $n_x \times n_x$, $n_x \times n_v$ and $n_y \times n_x$, respectively.

In this study, $F(\cdot)$ is represented by a PWL function. Note, in contrast to the previous studies on Hammerstein MPC [7,9,10], the nonlinear function $F(\cdot)$ is allowed to be non-invertible and coupled multivariable nonlinearities. The coupled multivariable nonlinearities mean that multiple outputs of $F(\cdot)$ are nonlinear functions of all inputs, which are often encountered in



$\boldsymbol{u}_k$
$\boldsymbol{v}_k = F(\boldsymbol{u}_k)$
$\boldsymbol{v}_k$
$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{v}_k$
$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k$
$\boldsymbol{y}_k$
Static nonlinear block
Dynamic linear block

Fig. 1 The discrete-time Hammerstein model

multivariable industrial processes [3]. In the processes with coupled multivariable nonlinearities, transforming input constraints to the constraints on immediate variables is not easy, which seriously limits the application of the previously proposed inversion-based Hammerstein MPC algorithms. This paper only assumes $F(\cdot)$ is Lipschitz continuous. Such an assumption is not restrictive because static description of numerous technological processes, e.g., chemical reactors or distillation columns, satisfies that condition.

2.2 Canonical piecewise linear function

In its typical form, a PWL function simply consists of a number of linear sub-functions, which are used only in their specific subregions [19]. Such classical representation requires an excessive number of parameters and may make identification very difficult. For this reason, Chua et al. [26] proposed a canonical expression (with minimum and necessary number of parameters) for PWL functions in $\mathbb{R}^1$ in the 1970s. It is until 1999 that Julian et al. [18,19] by using the so-called simplicial partition method, developed the canonical PWL representation in $\mathbb{R}^n$. The simplicial partition method is employed to divide the input domain to determine subregions, which has a wider range of application and is much simpler than other partition methods (e.g., the clustering method [27] and the prior knowledge-based method [28]). The resulting canonical PWL representation can approximate nonlinear mappings sufficiently closely, and the corresponding parameters can be calculated by the existing toolbox [20] directly and conveniently. In the following, the relevant definitions and concepts of the canonical PWL function are briefly reviewed. More details can be found in [18,19].

Definition 1 Simplex: Let $\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ be $n+1$ vectors in $\mathbb{R}^n$. A simplex (polytope) $\boldsymbol{S}$ is defined by

$$\boldsymbol{S} \triangleq \left\{ \boldsymbol{u} \,\middle|\, \boldsymbol{u} = \sum_{i=0}^{n} \mu_i \boldsymbol{u}_i \right\} \tag{3}$$

where $\boldsymbol{\mu} = [\mu_0\ \mu_1\ \ldots\ \mu_n]^{\mathrm{T}}$ is the simplex parameter vector, $0 \leq \mu_i \leq 1$ and $\sum_{i=0}^{n} \mu_i = 1$.

Consider a domain in $\mathbb{R}^n$ of the form

$$\boldsymbol{D} = \left\{ \boldsymbol{u} \in \mathbb{R}^n \,\middle|\, \underline{u}^i \leq u^i \leq \overline{u}^i, i = 1, 2, \ldots, n \right\} \tag{4}$$

where $\underline{u}^i$ and $\overline{u}^i$ are the lower bound and the upper bound of u^i, respectively. The simplicial partition of $\boldsymbol{D}$ consists of two steps. Firstly, by specifying the number of divisions (m_i) associated with the u^i axis (i.e., dividing the interval $[\underline{u}^i, \overline{u}^i]$ into m_i divisions), the domain $\boldsymbol{D}$ is evenly divided into $\prod_{i=1}^{n} m_i$ hypercubes

$$\begin{aligned} \boldsymbol{H_h} \triangleq & [\underline{u}^1 + h^1\delta_1,\ \underline{u}^1 + (h^1+1)\delta_1] \\ & \times [\underline{u}^2 + h^2\delta_2,\ \underline{u}^2 + (h^2+1)\delta_2] \\ & \vdots \\ & \times [\underline{u}^n + h^n\delta_n,\ \underline{u}^n + (h^n+1)\delta_n] \end{aligned} \tag{5}$$

where $\boldsymbol{h} = [h^1\ h^2\ \ldots\ h^n]^{\mathrm{T}}$, h^i are integers and $0 \le h^i \le m_i - 1$. In the axis u^i, the grid step is $\delta_i = (\overline{u}^i - \underline{u}^i)/m_i$. The corresponding grid step vector is formed as $\boldsymbol{\delta} = [\delta_1\ \delta_2\ \ldots\ \delta_n]^{\mathrm{T}}$. Each hypercube is characterized by 2^n vertices. In particular, the vertex $\boldsymbol{u_h} = [\underline{u}^1 + h^1\delta_1,\ \underline{u}^2 + h^2\delta_2,\ \ldots,\ \underline{u}^n + h^n\delta_n]^{\mathrm{T}}$ can uniquely characterize $\boldsymbol{H_h}$, and thus, $\boldsymbol{h}$ is used as the index for $\boldsymbol{H_h}$. In the second stage of the partition procedure, each hypercube is subdivided into $n!$ simplices $\boldsymbol{S}_s, s = 1, \ldots, n!$. Each simplex is characterized by $n+1$ vertices. All the simplices in the same hypercube $\boldsymbol{H_h}$ have the common vertex $\boldsymbol{u_h}$. By using the two-step simplicial partition, the domain $\boldsymbol{D}$ is finally divided into $n! \times \prod_{i=1}^{n} m_i$ regions (denoted as $\boldsymbol{H_h S}_s$). Partitioning of a domain in $\mathbb{R}^2$ is shown in Fig. 2.

Based on the simplicial partition, a PWL function is represented in the following canonical style

$$F(\boldsymbol{u}) = \boldsymbol{c}^{\mathrm{T}} \boldsymbol{\Lambda}(\boldsymbol{u}) \tag{6}$$

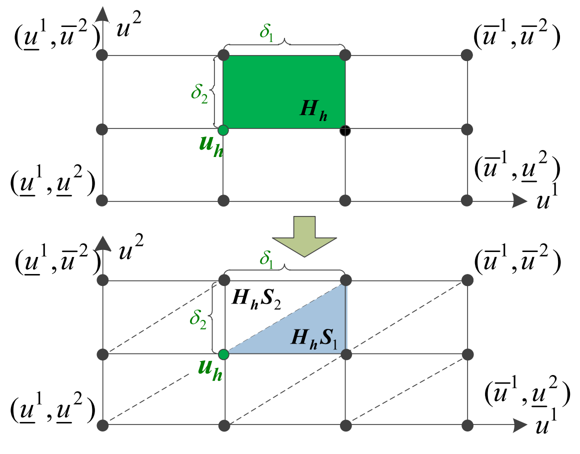

$(\underline{u}^1, \overline{u}^2)$
u^2
δ_1
$(\overline{u}^1, \overline{u}^2)$
δ_2
H_h
u_h
$(\underline{u}^1, \underline{u}^2)$
$(\overline{u}^1, \underline{u}^2)$
u^1
$(\underline{u}^1, \overline{u}^2)$
u^2
δ_1
$(\overline{u}^1, \overline{u}^2)$
H_hS_2
δ_2
H_hS_1
u_h
$(\underline{u}^1, \underline{u}^2)$
$(\overline{u}^1, \underline{u}^2)$
u^1

Fig. 2 The simplicial partition of a domain in $\mathbb{R}^2$

where $\boldsymbol{c}$ is the parameter vector of length $\prod_{i=1}^{n}(m_i+1)$ and $\boldsymbol{\Lambda}(\boldsymbol{u})$ is a vector function formed by a set of generating functions (multiple nestings of absolute value functions) [18,19].

3 Nonlinear MPC based on the PWL Hammerstein model

To achieve offset-free control, some mechanism to deal with potential process disturbance and unavoidable process model mismatch should be introduced in the designed MPC algorithm. The standard method is to augment the process state with the artificially introduced disturbance [4,29–34]. As a result, the original process model becomes the augmented state-space model and the resulting MPC algorithm can achieve offset-free control. It is also possible to use the non-minimal state-space model, in which the extended states consist of input increments, output increments and output signals [35–37]. Considering the augmented model method has the advantage of relatively low matrix dimensionality, it is used in this study. In the following, based on the PWL Hammerstein model defined by Eqs. (2), the MPC algorithm is developed by following the standard flow [29], i.e., the three parts: the augmented model design, the observer design, and the controller design.

3.1 Augmented model design

Considering the PWL Hammerstein model (2), the corresponding augmented model is designed as

$$\begin{aligned} & \boldsymbol{v}_k = F(\boldsymbol{u}_k) \\ & \begin{bmatrix} \boldsymbol{x}_{k+1} \\ \boldsymbol{d}_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B}_d \\ \boldsymbol{0}_{n_x \times n_d} & \boldsymbol{I}_{n_d \times n_d} \end{bmatrix}}_{\overline{\boldsymbol{A}}} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{d}_k \end{bmatrix} + \underbrace{\begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0}_{n_d \times n_v} \end{bmatrix}}_{\overline{\boldsymbol{B}}} \boldsymbol{v}_k + \boldsymbol{\zeta}_k \\ & \boldsymbol{y}_k = \underbrace{[\boldsymbol{C}\ \boldsymbol{C}_d]}_{\overline{\boldsymbol{C}}} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{d}_k \end{bmatrix} + \boldsymbol{\xi}_k \end{aligned} \tag{7}$$

where $\boldsymbol{d}_k \in \mathbb{R}^{n_d}$ is the augmented disturbance, $\boldsymbol{B}_d$ and $\boldsymbol{C}_d$ are the disturbance model matrices of dimensionality $n_x \times n_d$ and $n_y \times n_d$, respectively; $\overline{\boldsymbol{A}}$, $\overline{\boldsymbol{B}}$, and $\overline{\boldsymbol{C}}$ are the augmented model matrices of dimensionality $(n_x + n_d) \times (n_x + n_d)$, $(n_x + n_d) \times n_v$ and $n_y \times (n_x + n_d)$, respectively; $\boldsymbol{\zeta}_k = [\boldsymbol{\zeta}_{x,k}^{\mathrm{T}}\ \boldsymbol{\zeta}_{d,k}^{\mathrm{T}}]^{\mathrm{T}} \in$

$\mathbb{R}^{n_x+n_d}$ and $\boldsymbol{\xi}_k \in \mathbb{R}^{n_y}$ are zero-mean white noise for the augmented state and the output. The variances of $\boldsymbol{\zeta}_{x,k}$, $\boldsymbol{\zeta}_{d,k}$, and $\boldsymbol{\xi}_k$ are specified by matrices $\boldsymbol{Q}_x$, $\boldsymbol{Q}_d$, and $\boldsymbol{Q}_y$ of dimensionality $n_x \times n_x$, $n_d \times n_d$ and $n_y \times n_y$, respectively.

The disturbance model is of critical importance for offset-free control. Although the matrices $\boldsymbol{B}_d$ and $\boldsymbol{C}_d$ can be estimated from measured data, they are generally artificially selected for reducing modeling cost and obtaining good control performance. The most classical existing MPC algorithms for Hammerstein models [7,9,10,17,25] use only the output disturbance model in which $\boldsymbol{B}_d = \boldsymbol{0}_{n_x \times n_d}$, $\boldsymbol{C}_d = \boldsymbol{I}_{n_y \times n_d}$ with $n_d = n_y$. It is because MPC with such a simplified model may effectively compensate for a wide range of typical disturbances and model mismatch. However, the output disturbance model has two serious drawbacks as detailed in [32,33,38]. First, it cannot result in offset-free control if the controlled plant contains integration. Second, it often leads to sluggish closed-loop performance due to its poor ability to model actual disturbance dynamics. As an alternative to using the output disturbance model, the input disturbance one has been recommended in recent research for linear MPC algorithms [32,33,38] since it is likely to alleviate the aforementioned disadvantages. Because the nonlinear function $F(\cdot)$ is static, the input disturbance model is implemented by choosing $\boldsymbol{B}_d = \boldsymbol{B}$ and $\boldsymbol{C}_d = \boldsymbol{0}_{n_y \times n_d}$ with $n_d = n_u$. In this work, both input and output disturbance models are considered for controller design and finally one of them is chosen by comparing their control quality.

3.2 Observer design

After designing the augmented model, a proper observer is required to estimate the state $\boldsymbol{x}_k$ and the augmented disturbance $\boldsymbol{d}_k$ from the measured process input and output signals. An Augmented Kalman Filter (AKF) [32,33] can act as an observer to solve the estimation problem.

The AKF consists of the following two steps: model prediction and measurement correction. The first step is described by the equation

$$\begin{aligned}\boldsymbol{v}_k &= F(\boldsymbol{u}_k)\\ \begin{bmatrix}\boldsymbol{x}_{k|k-1}\\ \boldsymbol{d}_{k|k-1}\end{bmatrix} &= \overline{\boldsymbol{A}}\begin{bmatrix}\boldsymbol{x}_{k-1|k-1}\\ \boldsymbol{d}_{k-1|k-1}\end{bmatrix} + \overline{\boldsymbol{B}}\boldsymbol{v}_{k-1}\end{aligned} \tag{8}$$

whereas the second step is characterized by

$$\begin{bmatrix}\boldsymbol{x}_{k|k}\\ \boldsymbol{d}_{k|k}\end{bmatrix} = \begin{bmatrix}\boldsymbol{x}_{k|k-1}\\ \boldsymbol{d}_{k|k-1}\end{bmatrix} + \boldsymbol{L}_{\mathrm{KF}}\left(\boldsymbol{y}_k - \overline{\boldsymbol{C}}\begin{bmatrix}\boldsymbol{x}_{k|k-1}\\ \boldsymbol{d}_{k|k-1}\end{bmatrix}\right) \tag{9}$$

where $\boldsymbol{L}_{\mathrm{KF}} = [\boldsymbol{L}_x^{\mathrm{T}}\ \boldsymbol{L}_d^{\mathrm{T}}]^{\mathrm{T}}$ is the observer gain matrix of dimensionality $(n_x + n_d) \times n_y$. $\boldsymbol{L}_x^{\mathrm{T}}$ and $\boldsymbol{L}_d^{\mathrm{T}}$ are the state gain matrix and the augmented disturbance gain matrix of dimensionality $n_x \times n_y$ and $n_d \times n_y$, respectively. The gain matrix $\boldsymbol{L}_{\mathrm{KF}}$ can be computed offline by solving the following algebraic Riccati equation (ARE)

$$\boldsymbol{L}_{\mathrm{KF}} = \boldsymbol{P}\overline{\boldsymbol{C}}(\overline{\boldsymbol{C}}\boldsymbol{P}\overline{\boldsymbol{C}}^{\mathrm{T}} + \boldsymbol{Q}_y)^{-1} \tag{10}$$

where

$$\boldsymbol{P} = \overline{\boldsymbol{A}}\boldsymbol{P}\overline{\boldsymbol{A}}^{\mathrm{T}} - \overline{\boldsymbol{A}}\boldsymbol{P}\overline{\boldsymbol{C}}^{\mathrm{T}}(\overline{\boldsymbol{C}}\boldsymbol{P}\overline{\boldsymbol{C}}^{\mathrm{T}} + \boldsymbol{Q}_y)^{-1}\overline{\boldsymbol{C}}\boldsymbol{P}\overline{\boldsymbol{A}}^{\mathrm{T}} + \overline{\boldsymbol{Q}} \tag{11}$$

where $\overline{\boldsymbol{Q}} = \mathrm{diag}(\boldsymbol{Q}_x, \boldsymbol{Q}_d)$. The variance matrices $\boldsymbol{Q}_x$, $\boldsymbol{Q}_d$ and $\boldsymbol{Q}_y$ are treated as the adjustable observer parameters to balance the offset-free control speed and the control sensitivity to noises. Because $F(\cdot)$ is just a nonlinear function of the system input $\boldsymbol{u}_k$ rather than the estimated states, the above AKF (8) and (9) is essentially a linear Kalman filer, whose properties (e.g., the observation error convergence) have been well studied in previous literature [32,33] and thus is omitted here.

In addition to AKF, moving horizon estimation (MHE) [39] can also be used for state estimation. Different with AKF, MHE can explicitly deal with the constraints on the states to be estimated. On the other hand, MHE generally has a higher computational burden than AKF. If state constraints are available and the computational cost is acceptable, MHE may be alternatively used.

3.3 MPC optimization problem

The objective of the MPC algorithm is to minimize differences between the predefined reference trajectory and the predicted process outputs and to avoid excessive control actions. Thus, the future sequence of increments of the manipulated variables

$$\Delta\boldsymbol{U}_k = \left[\Delta\boldsymbol{u}_{k|k}^{\mathrm{T}}\ \Delta\boldsymbol{u}_{k+1|k}^{\mathrm{T}} \ldots \Delta\boldsymbol{u}_{k+Hc-1|k}^{\mathrm{T}}\right]^{\mathrm{T}}$$

is determined online at each sampling instant by solving the following optimization problem with constraints

$$
\begin{aligned}
\min_{\substack{\Delta \boldsymbol{u}_{k|k},\cdots \\ \Delta \boldsymbol{u}_{k+Hc-1|k}}} \quad & \sum_{i=k+1}^{k+Hp} \left\| \boldsymbol{y}^r_{i|k} - \boldsymbol{y}_{i|k} \right\|^2_{\boldsymbol{w}_y} + \sum_{i=k}^{k+Hc-1} \left\| \Delta \boldsymbol{u}_{i|k} \right\|^2_{\boldsymbol{w}_{\Delta u}} \\
\text{s.t.} \quad & \boldsymbol{y}_{i|k} = \boldsymbol{C}\boldsymbol{x}_{i|k} + \boldsymbol{C}_d \boldsymbol{d}_{i|k} \\
& \boldsymbol{x}_{i+1|k} = \boldsymbol{A}\boldsymbol{x}_{i|k} + \boldsymbol{B}_d \boldsymbol{d}_{i|k} + \boldsymbol{B}\boldsymbol{v}_{i|k} \\
& \boldsymbol{v}_{i|k} = F(\boldsymbol{u}_{i|k}) \\
& \boldsymbol{u}_{i|k} = \boldsymbol{u}_{i-1|k} + \Delta \boldsymbol{u}_{i|k} \\
& \Delta \boldsymbol{u}_{i|k} = 0, \text{ if } i \geq Hc \\
& \boldsymbol{y}_{\min} \leq \boldsymbol{y}_{i|k} \leq \boldsymbol{y}_{\max} \\
& \boldsymbol{u}_{\min} \leq \boldsymbol{u}_{i|k} \leq \boldsymbol{u}_{\max} \\
& \Delta \boldsymbol{u}_{\min} \leq \Delta \boldsymbol{u}_{i|k} \leq \Delta \boldsymbol{u}_{\max}
\end{aligned}
\tag{12}
$$

where the norm is defined as $\|\boldsymbol{x}\|^2_{\boldsymbol{M}} \triangleq \boldsymbol{x}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{x}$; $\boldsymbol{y}^r_{i|k} \in \mathbb{R}^{n_y}$ is the assumed reference trajectory for the sampling instant i known at the instant k; Hp is the prediction horizon; Hc is the control horizon; $\boldsymbol{w}_y \geq \boldsymbol{0}_{n_y \times n_y}$, $\boldsymbol{w}_{\Delta u} \geq \boldsymbol{0}_{n_u \times n_u}$ are weights of the minimized cost-function; $\boldsymbol{y}_{\min} \in \mathbb{R}^{n_y}$, $\boldsymbol{y}_{\max} \in \mathbb{R}^{n_y}$ are the constraints imposed on the system outputs; $\boldsymbol{u}_{\min} \in \mathbb{R}^{n_u}$, $\boldsymbol{u}_{\max} \in \mathbb{R}^{n_u}$, $\Delta\boldsymbol{u}_{\min} \in \mathbb{R}^{n_u}$, and $\Delta\boldsymbol{u}_{\max}$ are the constraints imposed on the magnitude and on the increments of system inputs, respectively. The output constraints may lead to infeasibility problems. In such cases, the so-called soft constraints can be used to replace the original output constraints, which is a standard approach in MPC. Readers can find more details in the classical textbooks [5,34,35].

At each sampling instant, n_uHc future control increments are calculated from the MPC optimization problem (12), then only the increments for the instant k are actually applied to the process, namely $\boldsymbol{u}_k = \boldsymbol{u}_{k-1} + \Delta\boldsymbol{u}_k$. At the next sampling instant, the whole optimization procedure is repeated.

It should be noted that, because the mapping from $\boldsymbol{u}_k$ to $\boldsymbol{v}_k$ (i.e., the function $F(\cdot)$) is essentially nonlinear, the predicted outputs ($\boldsymbol{y}_{i|k}$) are nonlinear functions of the calculated online control increments $\Delta\boldsymbol{U}_k$. This means that the future control increments are computed by solving the complex nonlinear optimization problem (12) [40]. In this paper, such a control algorithm is called as the MPC algorithm with nonlinear optimization (MPC-NL). The significant online computational burden seriously limits the application of the MPC-NL algorithm.

Bearing in mind possible computational complexity of the MPC-NL algorithm, this paper develops a computationally efficient MPC algorithm based on a multistep linearization technique (MPC-ML). At each sampling instant, by using an assumed input trajectory (multistep future inputs), the PWL functions are first transformed within the control horizon into multiple linear functions. Consequently, the predicted outputs become the linear functions of the future control increments, and the nonlinear optimization problem (12) is simplified into a QP problem, so that the future control increments can be easily computed (computational burden of quadratic optimization is much lower than that of general nonlinear optimization and the global optimal solution is always found). For achieving good approximation accuracy, the above-mentioned linearization and the resulting QP optimization may be repeated several times in internal iterations at each sampling instant.

3.4 Linearization of predicted trajectory

Taking into account the augmented Hammerstein model defined by Eq. (7), the predicted trajectory $\boldsymbol{V}_k$ of the intermediate variables between the static and dynamic parts of the model (over the control horizon) is

$$
\underbrace{\begin{bmatrix} \boldsymbol{v}_{k|k} \\ \boldsymbol{v}_{k+1|k} \\ \vdots \\ \boldsymbol{v}_{k+Hc-1|k} \end{bmatrix}}_{\boldsymbol{V}_k} = \begin{bmatrix} F(\boldsymbol{u}_{k-1} + \Delta\boldsymbol{u}_{k|k}) \\ F(\boldsymbol{u}_{k-1} + \Delta\boldsymbol{u}_{k+1|k}) \\ \vdots \\ F(\boldsymbol{u}_{k-1} + \Delta\boldsymbol{u}_{k+Hc-1|k}) \end{bmatrix}, \tag{13}
$$

Consequently, the predicted output trajectory $\boldsymbol{Y}_k$ (over the prediction horizon) is

$$
\underbrace{\begin{bmatrix} \boldsymbol{y}_{k+1|k} \\ \boldsymbol{y}_{k+2|k} \\ \vdots \\ \boldsymbol{y}_{k+Hc|k} \\ \vdots \\ \boldsymbol{y}_{k+Hp|k} \end{bmatrix}}_{\boldsymbol{Y}_k} = \underbrace{\begin{bmatrix} \boldsymbol{CB} & 0 & \cdots & 0 \\ \boldsymbol{CAB} & \boldsymbol{CB} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{CA}^{Hc-1}\boldsymbol{B} & \boldsymbol{CA}^{Hc-2}\boldsymbol{B} & \cdots & \boldsymbol{CB} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{CA}^{Hp-1}\boldsymbol{B} & \boldsymbol{CA}^{Hp-2}\boldsymbol{B} & \cdots & \sum_{i=0}^{Hp-Hc} \boldsymbol{CA}^i\boldsymbol{B} \end{bmatrix}}_{\boldsymbol{\Gamma}_v} \boldsymbol{V}_k
$$
$$
+ \underbrace{\begin{bmatrix} \boldsymbol{CA} \\ \boldsymbol{CA}^2 \\ \vdots \\ \boldsymbol{CA}^{Hc} \\ \vdots \\ \boldsymbol{CA}^{Hp} \end{bmatrix}}_{\boldsymbol{\Gamma}_x} \boldsymbol{x}_{k|k} + \underbrace{\begin{bmatrix} \boldsymbol{C}_d + \boldsymbol{CB}_d \\ \boldsymbol{C}_d + \sum_{i=0}^{1} \boldsymbol{CA}^i\boldsymbol{B}_d \\ \vdots \\ \boldsymbol{C}_d + \sum_{i=0}^{Hc-1} \boldsymbol{CA}^i\boldsymbol{B}_d \\ \vdots \\ \boldsymbol{C}_d + \sum_{i=0}^{Hp-1} \boldsymbol{CA}^i\boldsymbol{B}_d \end{bmatrix}}_{\boldsymbol{\Gamma}_d} \boldsymbol{d}_{k|k}. \tag{14}
$$

From Eq. (13) it is important to notice that the trajectory $\boldsymbol{V}_k$ is a nonlinear function of the calculated future control increments, i.e., $\Delta\boldsymbol{u}_{k+i|k}$ for $i = 0, \ldots, \mathrm{Hc}-1$, whereas from Eq. (14) one may notice that the output trajectory $\boldsymbol{Y}_k$ is linear in terms of the trajectory $\boldsymbol{V}_k$. If it would be possible to obtain a linear representation of the $\boldsymbol{V}_k$ trajectory with respect to $\Delta\boldsymbol{u}_{k+i|k}$, the resulting output trajectory becomes linear in terms of the calculated control increments. For this purpose, the function $F(\cdot)$ should be linearized and its linear approximation should be used for prediction.

The following three-step procedure is designed to linearize the function $F(\cdot)$ for a given input vector $\dot{\boldsymbol{u}} = [\dot{u}^1, \ldots, \dot{u}^n]^{\mathrm{T}}$ (a point in the domain $\boldsymbol{D}$).

Step 1 Find the hypercube containing $\dot{\boldsymbol{u}}$. As indicated in Eq. (5), a vector $\boldsymbol{h} = [h^1 \ \ldots \ h^n]^{\mathrm{T}}$ can uniquely index a hypercube $\boldsymbol{H_h}$. Hence, one can find the hypercube by computing the vector $\boldsymbol{h}$ from $\dot{\boldsymbol{u}}$. The specific procedure for completing Step 1 is:

(1a) Calculate $h^i = \mathrm{floor}((\dot{u}^i - \underline{u}^i)/\delta_i)$ $i = 1, \ldots, n$.
(1b) If $h^i = m_i$, let $h^i = m_i - 1$.
(1c) Repeat step (1a) and (1b) for $i = 1, \ldots, n$.

In Step 1, one can find a unique hypercube $\boldsymbol{H_h}$ for the given vector $\dot{\boldsymbol{u}}$ (which defines the current linearization point), even if $\dot{\boldsymbol{u}}$ is located at the intersection between hypercubes and the boundary of the domain $\boldsymbol{D}$.

Step 2 Find the simplex that $\dot{\boldsymbol{u}}$ belongs to.

Lemma 1 *Assume a simplex $\boldsymbol{H_h S}_s$ is characterized by $n+1$ vectors $\boldsymbol{u}_0, \ldots, \boldsymbol{u}_n$. The necessary and sufficient condition of $\dot{\boldsymbol{u}} \in \boldsymbol{H_h S}_s$ is there exists an unique simplex parameter vector $\hat{\boldsymbol{\mu}} = [\hat{\mu}_0 \ \hat{\mu}_1 \ \ldots \ \hat{\mu}_n]^{\mathrm{T}}$ subject to the condition*

$$\begin{cases} \hat{\boldsymbol{\mu}} = \underbrace{\begin{bmatrix} \boldsymbol{u}_0 & \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_n \\ 1 & 1 & \cdots & 1 \end{bmatrix}^{-1}}_{S_u^{-1}} \begin{bmatrix} \dot{\boldsymbol{u}} \\ 1 \end{bmatrix} \\ 0 \le \hat{\mu}_i \le 1, \quad i = 1, \ldots, n. \end{cases} \tag{15}$$

*The proof of Lemma 1 is straightforward from the definition of a simplex (**Definition 1**).*

By using the condition (15), one can check the simplices in the hypercube $\boldsymbol{H_h}$ one by one. However, it should be noted that the inversion of the matrix $\boldsymbol{S}_u$

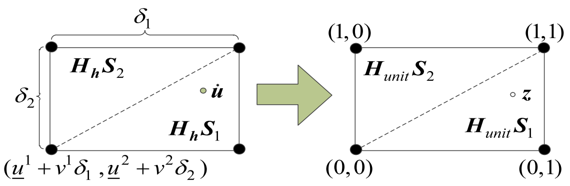

δ_1
δ_2
$\boldsymbol{H_h S}_2$
$\dot{\boldsymbol{u}}$
$\boldsymbol{H_h S}_1$
$(\underline{u}^1 + v^1\delta_1, \underline{u}^2 + v^2\delta_2)$
(1,0)
(1,1)
$\boldsymbol{H}_{unit}\boldsymbol{S}_2$
z
$\boldsymbol{H}_{unit}\boldsymbol{S}_1$
(0,0)
(0,1)

Fig. 3 The translation T from $\boldsymbol{H_h}$ to $\boldsymbol{H}_{\mathrm{unit}}$ in $\mathbb{R}^2$

has to be calculated online because $\boldsymbol{S}_u$ changes with $\dot{\boldsymbol{u}}$, which will lead to unnecessary computing cost.

A linear transformation $T : \mathbb{R}^n \to \mathbb{R}^n$ is defined as

$$z = T(\dot{\boldsymbol{u}}) = (\dot{\boldsymbol{u}} - \underline{\boldsymbol{u}})./\boldsymbol{\delta} - \boldsymbol{h} \tag{16}$$

where './' represents the element-wise division.

The transformation T is employed to "translate" the hypercube $\boldsymbol{H_h}$ into the unit hypercube $\boldsymbol{H}_{\mathrm{unit}} = [0, 1]^n$ which is illustrated in Fig. 3 for $\mathbb{R}^2$. It is obvious that the hypercube $\boldsymbol{H}_{\mathrm{unit}}$ has the same number of simplices as $\boldsymbol{H_h}$ has. Moreover, the relative locations of the simplices in the same hypercube do not change after the linear transformation. For example, both $\boldsymbol{H_h S}_1$ and $\boldsymbol{H}_{\mathrm{unit}}\boldsymbol{S}_1$ are located at the bottom left in the corresponding hypercubes. In this regard, one can determine the simplex containing $\dot{\boldsymbol{u}}$ by judging which simplex in the unit hypercube $\boldsymbol{H}_{\mathrm{unit}}$ includes z.

Lemma 2 *If and only if the following condition is satisfied*

$$\begin{cases} \hat{\boldsymbol{\mu}} = \underbrace{\begin{bmatrix} z_0 & z_1 & \cdots & z_n \\ 1 & 1 & \cdots & 1 \end{bmatrix}^{-1}}_{S_z^{-1}} \begin{bmatrix} z \\ 1 \end{bmatrix} \\ 0 \le \hat{\mu}_i \le 1, \quad i = 1, \ldots, n \end{cases} \tag{17}$$

where $z_i = T(\boldsymbol{u}_i)$, $i = 0, \ldots, n$ and $z = T(\dot{\boldsymbol{u}})$, then $\dot{\boldsymbol{u}} \in \boldsymbol{H_h S}_s$.

Unlike $\mathbf{S}_u^{-1}$, $\mathbf{S}_z^{-1}$ can be calculated offline and stored and can be used online in a look-up table manner.

The specific procedure for completing Step 2 is:

(2a) Compute $z = T(\dot{\boldsymbol{u}})$.
(2b) For each $\boldsymbol{H}_{\mathrm{unit}}\boldsymbol{S}_s$: if the condition (17) is satisfied, then $\dot{\boldsymbol{u}}$ is in $\boldsymbol{H}_h\boldsymbol{S}_s$; otherwise repeat checking the condition (17) for other simplices in $\boldsymbol{H}_{\mathrm{unit}}$ until the simplex including $\dot{\boldsymbol{u}}$ is found.

Step 3 Linearize the PWL function $F(\cdot)$. In each simplex $\boldsymbol{H_h S}_s$, the PWL function is equivalent to the following linear function

$$\begin{aligned} F(\boldsymbol{u}) &= F(\tilde{\boldsymbol{u}}_{\boldsymbol{h}}) + \left.\frac{\partial F}{\partial u^1}\right|_{\boldsymbol{h},s} (u^1 - \tilde{u}^1_{\boldsymbol{h}}) + \cdots \\ &\quad + \left.\frac{\partial F}{\partial u^n}\right|_{\boldsymbol{h},s} (u^n - \tilde{u}^n_{\boldsymbol{h}}) \\ &= \boldsymbol{\phi u} + \boldsymbol{\psi} \end{aligned} \tag{18}$$

where $\tilde{\boldsymbol{u}}_{\boldsymbol{h}} = [\tilde{u}^1_{\boldsymbol{h}} \ \ldots \ \tilde{u}^n_{\boldsymbol{h}}]$ is an arbitrary point in $\boldsymbol{H_h S}_s$ (typically $\boldsymbol{u_h}$); $\boldsymbol{\phi} = \left[\left.\frac{\partial F}{\partial u^1}\right|_{\boldsymbol{h},s} \cdots \left.\frac{\partial F}{\partial u^n}\right|_{\boldsymbol{h},s} \right]$; $\boldsymbol{\psi} = F(\tilde{\boldsymbol{u}}_{\boldsymbol{h}}) - \boldsymbol{\phi} \tilde{\boldsymbol{u}}_{\boldsymbol{h}}$. Note that given an identified PWL function, $\boldsymbol{\phi}$ in different simplices can be also calculated offline and stored for online use. The specific procedure for completing Step 3 is:

(3a) Find $\boldsymbol{\phi}$.
(3b) Calculate $\boldsymbol{\psi}$.

The above 3-step procedure can be used to linearize the function $F(\cdot)$ at a single vector $\dot{\boldsymbol{u}}$. In order to linearize the trajectory $\boldsymbol{V}_k$ and finally linearize the predicted output trajectory $\boldsymbol{Y}_k$, the linearization is performed for some trajectory of the manipulated variables defined over the whole control horizon, which is defined in the following way

$$\dot{\boldsymbol{U}}^t_k = \left[(\dot{\boldsymbol{u}}^t_{k|k})^{\mathrm{T}}, \ (\dot{\boldsymbol{u}}^t_{k+1|k})^{\mathrm{T}} \ \ldots \ (\dot{\boldsymbol{u}}^t_{k+Hc-1|k})^{\mathrm{T}} \right]^{\mathrm{T}} \tag{19}$$

The trajectory of future values of the manipulated variables (19) of course depends on the current operating point of the process, i.e., it depends on the sampling time k. Moreover, at the same sampling instant it is possible to repeat linearization a few times, the index t indicates the internal iteration (repetition of linearization).

By repeatedly using the 3-step procedure, the trajectory of the intermediate variables between the static and dynamic parts of the model (over the control horizon), $\boldsymbol{V}_k$, is linearized around the trajectory of the future process inputs (over the control horizon), $\dot{\boldsymbol{U}}^t$. From Eq. (18), the linearized trajectory is

$$\widetilde{\boldsymbol{V}}^t = \underbrace{\begin{bmatrix} \boldsymbol{\phi}^t_{k|k} & 0 & \cdots & 0 \\ 0 & \boldsymbol{\phi}^t_{k+1|k} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{\phi}^t_{k+Hc-1|k} \end{bmatrix}}_{\boldsymbol{\Phi}^t_k} \boldsymbol{U}^t + \underbrace{\begin{bmatrix} \boldsymbol{\psi}^t_{k|k} \\ \boldsymbol{\psi}^t_{k+1|k} \\ \vdots \\ \boldsymbol{\psi}^t_{k+Hc-1|k} \end{bmatrix}}_{\boldsymbol{\Psi}^t_k} \tag{20}$$

Considering that $\boldsymbol{U}^t_k = \boldsymbol{L}\Delta\boldsymbol{U}^t_k + \boldsymbol{U}_{k-1}$ where the matrix $\boldsymbol{L}$, of dimensionality $n_u Hc \times n_u Hc$, and the vector $\boldsymbol{U}_{k-1}$, of length $n_u Hc$, are

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{I}_{n_u \times n_u} & \boldsymbol{0}_{n_u \times n_u} & \cdots & \boldsymbol{0}_{n_u \times n_u} \\ \boldsymbol{I}_{n_u \times n_u} & \boldsymbol{I}_{n_u \times n_u} & \cdots & \boldsymbol{0}_{n_u \times n_u} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{I}_{n_u \times n_u} & \boldsymbol{I}_{n_u \times n_u} & \cdots & \boldsymbol{I}_{n_u \times n_u} \end{bmatrix}, \boldsymbol{U}_{k-1} = \begin{bmatrix} \boldsymbol{u}_{k-1} \\ \boldsymbol{u}_{k-1} \\ \vdots \\ \boldsymbol{u}_{k-1} \end{bmatrix}$$

the linearized trajectory (20) may be compactly expressed

$$\widetilde{\boldsymbol{V}}^t_k = \boldsymbol{\Phi}^t_k \boldsymbol{L}\Delta\boldsymbol{U}^t_k + \boldsymbol{\Phi}^t_k \boldsymbol{U}_{k-1} + \boldsymbol{\Psi}^t_k \tag{21}$$

Next, taking into account Eq. (14), the linearized trajectory of output predictions is

$$\widetilde{\boldsymbol{Y}}^t_k = \boldsymbol{\Gamma}_v \boldsymbol{\Phi}^t_k \boldsymbol{L}\Delta\boldsymbol{U}^t_k + \boldsymbol{Y}^{0,t}_k \tag{22}$$

where the so-called free trajectory (independent of the currently calculated future control sequence) is defined as

$$\boldsymbol{Y}^{0,t}_k = \boldsymbol{\Gamma}_v \boldsymbol{\Phi}^t_k \boldsymbol{U}_{k-1} + \boldsymbol{\Gamma}_v \boldsymbol{\Psi}^t_k + \boldsymbol{\Gamma}_x \boldsymbol{x}_{k|k} + \boldsymbol{\Gamma}_d \boldsymbol{d}_{k|k} \tag{23}$$

3.5 Efficient MPC algorithm

Taking into account the linear approximation of the predicted output trajectory defined by Eq. (22), the general nonlinear MPC optimization problem (12) becomes the following QP problem

$$\begin{aligned} \min_{\Delta\boldsymbol{U}^t_k} \quad & \left\| \boldsymbol{Y}^r_k - \boldsymbol{\Gamma}_v \boldsymbol{\Phi}^t_k \boldsymbol{L}\Delta\boldsymbol{U}^t_k - \boldsymbol{Y}^{0,t}_k \right\|^2_{\boldsymbol{W}_y} + \left\| \Delta\boldsymbol{U}^t \right\|^2_{\boldsymbol{W}_{\Delta u}} \\ s.t. \quad & \boldsymbol{Y}_l \le \boldsymbol{\Gamma}_v \boldsymbol{\Phi}^t_k \boldsymbol{L}\Delta\boldsymbol{U}^t_k + \boldsymbol{Y}^{0,t}_k \le \boldsymbol{Y}_u \\ & \boldsymbol{U}_l \le \boldsymbol{U}_{k-1} + \boldsymbol{L}\Delta\boldsymbol{U}^t_k \le \boldsymbol{U}_u \\ & \Delta\boldsymbol{U}_l \le \Delta\boldsymbol{U}^t_k \le \Delta\boldsymbol{U}_u \end{aligned} \tag{24}$$

where the vectors of length $n_y Hc$ are

$$\boldsymbol{Y}_k^r = \begin{bmatrix} \boldsymbol{y}_{k+1|k}^r \\ \boldsymbol{y}_{k+2|k}^r \\ \vdots \\ \boldsymbol{y}_{k+Hp|k}^r \end{bmatrix}, \; \boldsymbol{Y}_l = \begin{bmatrix} \boldsymbol{y}_{\min} \\ \boldsymbol{y}_{\min} \\ \vdots \\ \boldsymbol{y}_{\min} \end{bmatrix}, \; \boldsymbol{Y}_u = \begin{bmatrix} \boldsymbol{y}_{\max} \\ \boldsymbol{y}_{\max} \\ \vdots \\ \boldsymbol{y}_{\max} \end{bmatrix}$$

and the vectors of length $n_u Hp$ are

$$\boldsymbol{U}_l = \begin{bmatrix} \boldsymbol{u}_{\min} \\ \boldsymbol{u}_{\min} \\ \vdots \\ \boldsymbol{u}_{\min} \end{bmatrix}, \; \boldsymbol{U}_u = \begin{bmatrix} \boldsymbol{u}_{\max} \\ \boldsymbol{u}_{\max} \\ \vdots \\ \boldsymbol{u}_{\max} \end{bmatrix},$$

$$\Delta\boldsymbol{U}_l = \begin{bmatrix} \Delta\boldsymbol{u}_{\min} \\ \Delta\boldsymbol{u}_{\min} \\ \vdots \\ \Delta\boldsymbol{u}_{\min} \end{bmatrix}, \; \Delta\boldsymbol{U}_u = \begin{bmatrix} \Delta\boldsymbol{u}_{\max} \\ \Delta\boldsymbol{u}_{\max} \\ \vdots \\ \Delta\boldsymbol{u}_{\max} \end{bmatrix}$$

To sum up, the proposed MPC-ML algorithm consists of the following steps, which are repeated at each sampling instant k:

1. The state vector $\boldsymbol{x}_{k|k}$ and the disturbance vector $\boldsymbol{d}_{k|k}$ are estimated.
2. The vector which defines the linearization point is initialized, $\dot{\boldsymbol{U}}_k^1 = \left[\boldsymbol{u}_{k-1}^{\mathrm{T}}\ \boldsymbol{u}_{k-1}^{\mathrm{T}} \ldots \boldsymbol{u}_{k-1}^{\mathrm{T}}\right]^{\mathrm{T}}$, the index t is set to 1.
3. The linearized predicted output trajectory $\widetilde{\boldsymbol{Y}}_k^t$ is calculated, i.e., the matrices $\boldsymbol{\Phi}_k^t$ and $\boldsymbol{\Psi}_k^t$ are found.
4. The MPC QP problem (24) is solved to find $\Delta\boldsymbol{U}_k^t$ and the vector $\dot{\boldsymbol{U}}_k^t = \boldsymbol{U}_{k-1} + \boldsymbol{L}\Delta\boldsymbol{U}_k^t$ is updated.
5. Decide whether internal iterations should be terminated. If the trajectory $\Delta\boldsymbol{U}_k^t$ is close to the trajectory at the previous internal iteration, i.e.,

   $$\left\|\Delta\boldsymbol{U}_k^t - \Delta\boldsymbol{U}_k^{t-1}\right\|^2 < TV_u \tag{25}$$

   or $t > t_{\max}$, then go to Step 6; (TV_u is a threshold value to be tuned). Otherwise, update $t := t+1$ and go to Step 3.
6. Apply the first n_u elements of the vector $\Delta\boldsymbol{U}_k^t$ (i.e., the vector $\Delta\boldsymbol{u}_{k|k}^t$) to the process.

At the next sampling instant, the algorithm starts from Step 1 and the above six-step procedure is repeated.

Remark 1 A similar method, which is called MPC with nonlinear prediction and linearization along predicted trajectory (MPC-NPLPT), has been proposed for different types of process models, including Hammerstein [17,25], Wiener [5], Hammerstein-Wiener [41] as well as Wiener-Hammerstein cascade structures [42]. The MPC-NPLPT algorithm requires the process model to be differentiable and calculates the derivatives online, while the proposed MPC-ML algorithm linearizes the PWL function in the form of a look-up table. Moreover, the proposed MPC-ML can integrate various disturbance models, and thus can contribute to better control performance than MPC-NPLPT.

4 Simulation experiments

In this section, an single-input single-output (SISO) CSTR and a multiple-input multiple-output (MIMO) pH neutralization reactor are studied to show the advantages of the proposed MPC-ML algorithm.

First of all, the Hammerstein models are identified by the linear-nonlinear (L-N) approach [27,43]. This approach is straightforward and ensures an accurate estimation of the static nonlinearity. In the first step of the L-N approach, the linear block of the Hammerstein model is identified using the dynamic training data (the signals $\boldsymbol{u}$ and $\boldsymbol{y}$) and the 'n4sid' algorithm [44]. Then, the intermediate variable vector $\boldsymbol{v}$ is calculated using the process output of the steady-state training data and the static gain matrix of the linear model ($\boldsymbol{K}_{\mathrm{lin}}$) from the equation $\boldsymbol{v} = \boldsymbol{K}_{\mathrm{lin}}^{-1} \times \boldsymbol{y}$. Finally, the nonlinear PWL block of the model is found from the steady-state training data (the signals $\boldsymbol{v}$ and $\boldsymbol{y}$) using the PWL toolbox [20]. Multiple PWL Hammerstein models with different numbers of divisions are firstly identified, and then they are evaluated and selected by an independent dynamic test data set.

Based on the identified Hammerstein models, three types of MPC controllers, i.e., linear MPC (MPC-L), MPC based on a multistep linearization technique (MPC-ML) and MPC algorithm with nonlinear optimization (MPC-NL) are designed. In the above-mentioned algorithms, the same Hammerstein models are used as well as the same input and output disturbance models. For comparison, a MPC algorithm with nonlinear prediction and linearization along predicted trajectory (MPC-NPLPT) [41] is also designed based on the differentiable semiempirical Hammerstein process models [45]. Because the input disturbance model is not applicable in MPC-NPLPT, only the output disturbance model is used in this algorithm. Finally, the resulting MPC controllers are compared to show the advantages of the proposed MPC-ML algorithm. All the simulations are carried out using the software MATLAB R2015b in an Intel Core i5-4590, 3.30GHz computer with a 64-bit Windows 7 operating system.

4.1 Example 1: an SISO CSTR

Considering an SISO continuous stirred tank process, which consists of irreversible, exothermic reaction, $A \rightarrow B$, in a constant volume reactor cooled by a single coolant stream (see Fig. 4). It can be modeled by the following continuous-time nonlinear equations [11]

$$\begin{cases} \dot{C}_A = \frac{q}{V}(C_{A0} - C_A) - k_0 C_A e^{-\frac{E}{RT}} \\ \dot{T} = \frac{q}{V}(T_0 - T) - \frac{\Delta H k_0}{\rho C_p} C_A e^{-\frac{E}{RT}} \\ \quad + \frac{\rho_c C_{pc}}{\rho C_p V} q_c \left(1 - e^{-\frac{h_A}{\rho_c C_{pc} q_c}}\right)(T_{c0} - T) \end{cases} \tag{26}$$

The process output variable is the concentration of A, denoted by C_A. The process input variable is the coolant flow rate, denoted by q_c. The variation ranges of the output and the input are $C_A \in [0.02, 0.15]$ and $q_c \in [60, 115]$, respectively. The control objective is to regulate C_A for set-point tracking by manipulating q_c. The nominal model parameters are listed in Table 1. The sampling period is $T_s = 0.1$ min.

4.1.1 Modeling of the CSTR

At first, two training data sets, i.e., the dynamic training set and the steady-state one, are generated to identify the dynamic linear part and the static PWL part, respectively. The first set consists of 2000 samples around the nominal set-point, whereas the second one consists of 220 equidistant steady-state data points in the range $q_c \in [60, 115]$. Next, a dynamic test set with 1000 samples in the range $q_c \in [60, 115]$ is also generated for model evaluation and model selection. All the training data sets and the test data set are scaled by the nominal process input (103.41 L min^{-1}) and process output (0.1 mol L^{-1}): $u = q_c - 103.41$, $y = C_A - 0.1$.

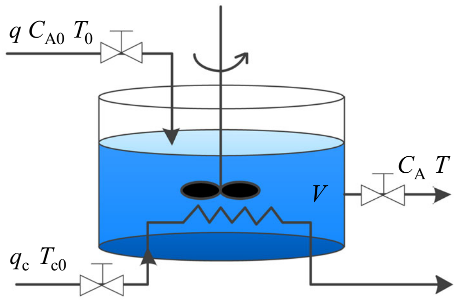


Fig. 4 The SISO continuous stirred tank reactor

Table 1 Nominal model parameters of CSTR

Measured product concentration	C_A	0.1 mol/L
Reactor temperature	T	438.5 K
Coolant flow rate	q_c	103.41 L/min
Process flow rate	q	100 L/min
Feed concentration	C_{A0}	1 mol/L
Feed temperature	T_0	350 K
Inlet coolant temperature	T_{c0}	350 K
CSTR Volume	V	100 L
Heat transfer term	h_A	7×10^5 cal/(min K)
Reaction rate constant	k_0	7.2×10^{10} min^{-1}
Activation energy term	E/R	1×10^4 K
Heat of reaction	ΔH	-2×10^5 cal/min
Liquid densities	ρ, ρ_c	1×10^3 g/L
Specific Heats	C_p, C_{pc}	1 ca/(g K)

The linear model is identified by the 'n4sid' algorithm [44]. The order of the linear model is determined by 4 ($n_x = 4$) as in [40]. Next, the static PWL model is found. It is interesting to consider the influence of the number of divisions (m) used in the static part of the model on the overall accuracy of the whole model. Total nine PWL functions are calculated, whose number of divisions (m) are 2, 3, . . . 10, respectively. Figure 5 depicts the steady-state data set versus the outputs of four PWL functions.

By combining the PWL functions and the linear model, the PWL Hammerstein models are finally formed. The mean squared errors (MSE$_{\text{test}}$) for the test data set are calculated to evaluate the identified Hammerstein models

$$\text{MSE}_{\text{test}} = \frac{1}{n_{\text{test}}} \sum_{k=1}^{n_{\text{test}}} (C_{A,k} - C_{A,k}^{\text{mod}})^2 \tag{27}$$

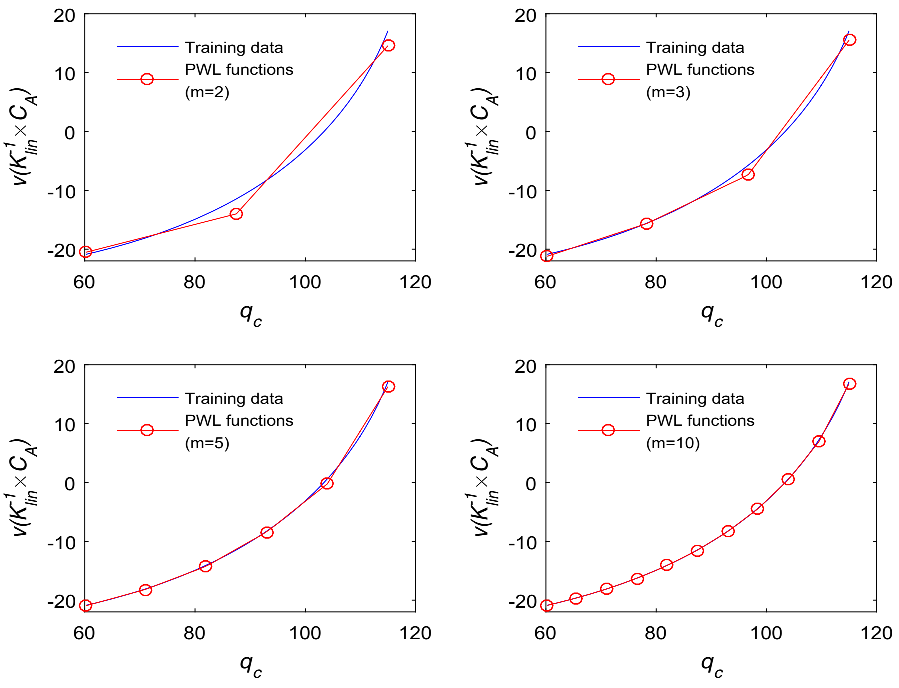


Fig. 5 Modeling of the CSTR: the training steady-state data set versus the obtained output of the PWL functions with different numbers of divisions

where n_{test} is the number of samples in the test set, $C_{A,k}$ denotes the real process output, $C_{A,k}^{\text{mod}}$ is the output of the models. Table 2 shows model test errors for the linear and PWL Hammerstein models with diffident numbers of divisions. The outputs of the PWL Hammerstein models with different numbers of divisions are depicted in Fig. 6. Additionally, the output of the linear model of the same order of dynamics is also shown. It is easily to notice that in general the PWL Hammerstein models are much more precise than the linear one. Finally, as a compromise between model complexity and accuracy, the model with less than six divisions ($m \leq 5$) is used in the MPC-ML algorithms.

Table 2 Comparison of linear and PWL Hammerstein models of CSTR

Model	m	MSE_{test}
Linear	–	1.1311×10^{-3}
PWL Hammerstein	2	1.8400×10^{-4}
PWL Hammerstein	3	1.0979×10^{-4}
PWL Hammerstein	4	9.0949×10^{-5}
PWL Hammerstein	5	8.5112×10^{-5}
PWL Hammerstein	6	9.0115×10^{-5}
PWL Hammerstein	7	9.1333×10^{-5}
PWL Hammerstein	8	9.2411×10^{-5}
PWL Hammerstein	9	9.2107×10^{-5}
PWL Hammerstein	10	8.7813×10^{-5}

4.1.2 Predictive control of the CSTR

To evaluate the influence of the number of divisions (m) on control performance, the PWL Hammerstein models with 2, 3, 4, and 5 divisions are used in the MPC-ML controllers. The most accurate PWL Hammerstein model, the one with 5 divisions, is used in the MPC-NL controllers. All MPC controllers have the same parameters $Hp = 100$, $Hc = 10$, $\boldsymbol{w}_y = 1000$ and $\boldsymbol{w}_{\Delta u} = 1$. Additional parameters of the MPC-ML controller are: $TV_u = 0.1$, $t_{\max} = 3$. Constraints on the manipulated variables are: $60 \text{ L min}^{-1} \leq q_c \leq 115 \text{ L min}^{-1}$.

Figures 7 and 8 show the closed-loop responses of CSTR for a chosen trajectory of set-point, with input and output disturbance models, respectively. In order to compare the MPC algorithms, the following integral square error (ISE) criterion is defined

$$\text{ISE} = \sum_{k=1}^{400} (C_{A,k}^r - C_{A,k})^2 \tag{28}$$

where $C_{A,k}^r$ represents the set-points. Table 3 compares the MPC controllers in terms of the ISE indicator and the scaled computation time (in relation to the most

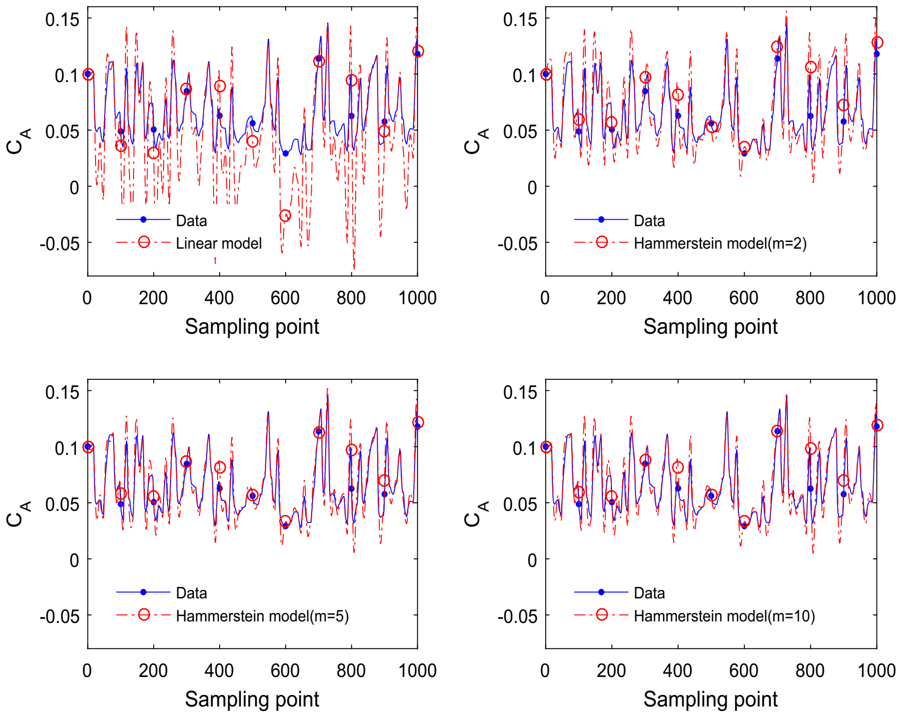


Fig. 6 Modeling of the CSTR: the linear model versus the Hammerstein models with different numbers of divisions

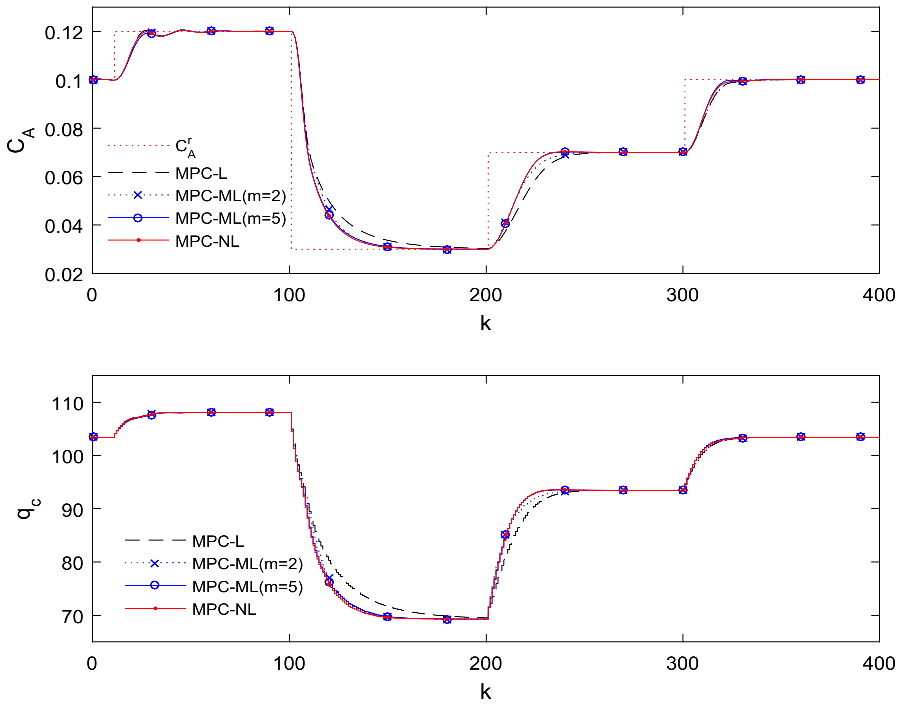


Fig. 7 Closed-loop responses of CSTR with the input disturbance model

computationally demanding MPC-NL scheme with the output disturbance model, whose computation time is 191.53s).

It can be seen that the MPC-ML controllers, the MPC-NPLPT controller (only applicable to the output disturbance model), and the MPC-NL controllers

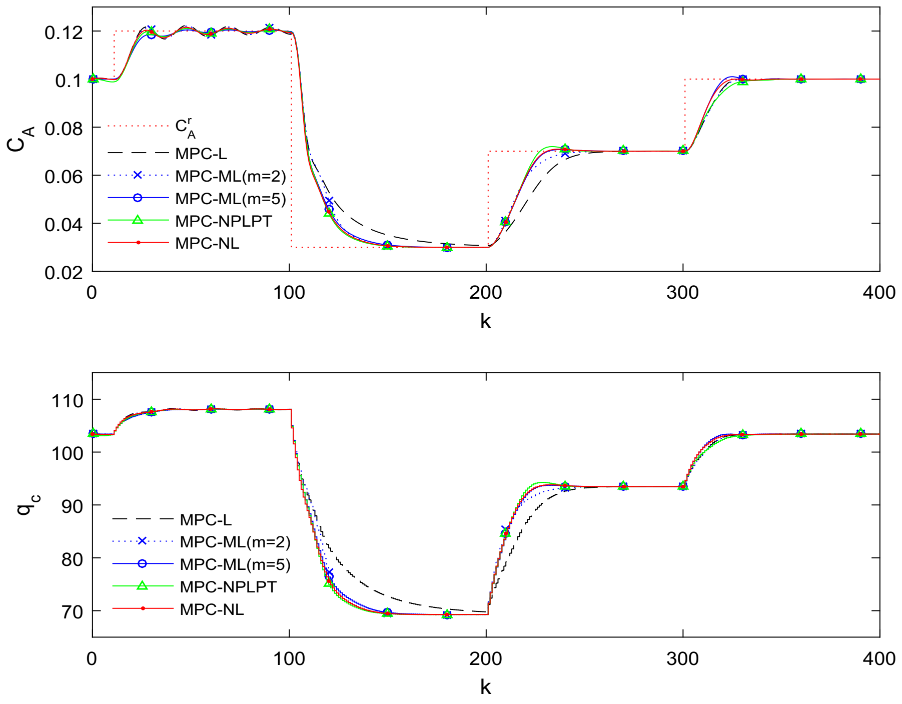


Fig. 8 Closed-loop responses of CSTR with output disturbance model

Table 3 Control accuracy and computational burden for the CSTR

Algorithm	m	Disturbance	ISE	Scaled time (%)
MPC-L	–	Input	9.30×10^{-2}	2.69
MPC-ML	2	Input	8.63×10^{-2}	4.57
MPC-ML	3	Input	8.24×10^{-2}	4.39
MPC-ML	4	Input	8.23×10^{-2}	4.52
MPC-ML	5	Input	8.20×10^{-2}	4.31
MPC-NL	5	Input	8.09×10^{-2}	99.96
MPC-L	–	Output	9.93×10^{-2}	2.62
MPC-ML	2	Output	8.93×10^{-2}	4.73
MPC-ML	3	Output	8.44×10^{-2}	4.58
MPC-ML	4	Output	8.40×10^{-2}	4.42
MPC-ML	5	Output	8.32×10^{-2}	4.67
MPC-NPLPT	–	Output	8.30×10^{-2}	23.54
MPC-NL	5	Output	8.18×10^{-2}	100.00

have similar closed-loop performance, both of which perform better than the MPC-L controllers. Because the identified linear model is only effective for the nominal working point, the MPC-L controllers give unsatisfactory sluggish tracking responses as the set-point being far away from the nominal working point ($C_A = 0.1$). By comparison, the nonlinear PWL Hammerstein model is effective in the entire working range. Besides, for this CSTR process, the MPC-ML based on more divisions (larger m) contribute to higher control accuracy than those with less divisions (smaller m). Meanwhile, the computation time of the MPC-ML controllers is far less than that for the MPC-NPLPT controller and the MPC-NL controllers, and is just slightly longer than that for the MPC-L controllers. Therefore, compared with other MPC controllers, the proposed MPC-ML can achieve a better balance between control performance and computational efficiency.

Moreover, the simulation results show all the controllers based on the input disturbance model perform better than their counterparts based on the output disturbance model. This illustrates the necessity of trying the input disturbance model and demonstrates the another advantage of MPC-ML over the traditional MPC-NPLPT (MPC-ML has the ability to integrate various disturbance model).

Based on the resulting control performance, the MPC-ML based on the input disturbance model and five divisions is recommended for the CSTR.

4.2 Example 2: a pH neutralization reactor

The pH neutralization reactor is shown in Fig. 9, in which the neutralization reaction takes place among acid (HNO_3), buffer ($NaHCO_3$), and base (NaOH). This process can be described by the following three continuous-time nonlinear differential equations [25, 46]

$$\begin{cases} \dot{W}_{a4} = \dfrac{1}{Ah}[(W_{a_1} - W_{a_4})q_1 + (W_{a_2} - W_{a_4})q_2 \\ \qquad + (W_{a_3} - W_{a_4})q_3] \\ \dot{W}_{b4} = \dfrac{1}{Ah}[(W_{b_1} - W_{b_4})q_1 + (W_{b_2} - W_{b_4})q_2 \\ \qquad + (W_{b_3} - W_{b_4})q_3] \\ \dot{h} = \dfrac{1}{A}(q_1 + q_2 + q_3 - c_v h^{0.5}) \end{cases} \tag{29}$$

and the nonlinear algebraic output equation

$$W_{a_4} + 10^{\mathrm{pH}-14} - 10^{-\mathrm{pH}} + W_{b_4} \times \frac{1 + 2 \times 10^{\mathrm{pH}-\mathrm{p}K_2}}{1 + 10^{\mathrm{p}K_1-\mathrm{pH}} + 10^{\mathrm{pH}-\mathrm{p}K_2}} = 0 \tag{30}$$

where q_i, W_{a_i}, and W_{b_i} are the flows, the charge balance factors, and the material balance factors, respectively; pK_1 and pK_2 are constant chemical coefficients. The control objective is to regulate the pH value of the effluent stream to predefined values and keep the liquid level h constant at the nominal value, i.e., $\boldsymbol{y} = [\mathrm{pH}\ h]^{\mathrm{T}}$. The manipulated variables are the acid flow q_1 and the base flow q_3, i.e., $\boldsymbol{u} = [q_1\ q_3]^{\mathrm{T}}$. The operating range of the manipulated variables is: $q_1 \in [0, 30]$ and $q_3 \in [0, 30]$.

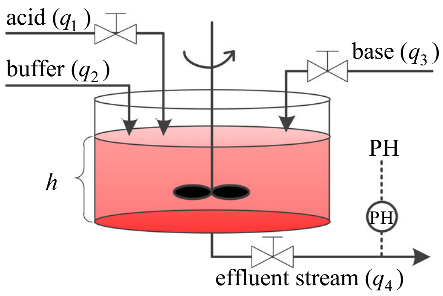


Fig. 9 The pH neutralization reactor

Table 4 Nominal model parameters of pH reactor

pH = 7.0255	$h = 14.0090$ cm
$q_1 = 16.6$ mL/s	$q_2 = 0.55$ mL/s
$q_3 = 15.6$ mL/s	$W_{a_1} = -4.32 \times 10^{-4}$ mol/L
$W_{a_2} = -3 \times 10^{-2}$ mol/L	$W_{a_3} = 5.28 \times 10^{-4}$ mol/L
$W_{b_1} = 0$	$W_{b_2} = 3 \times 10^{-2}$ mol/L
$W_{b_3} = 5 \times 10^{-5}$ mol/L	$A = 207$ cm^2
$c_v = 8.75$ mL/(cm s)	p$K_1 = 6.35$
p$K_2 = 10.25$	

The nominal parameters are listed in Table 4. The sampling period is $T_s = 10$s.

4.2.1 Modeling of the pH reactor

To model the pH reactor, the similar procedures as in Example 1 are performed. Two training sets, i.e., the dynamic training set and the steady-state one, are generated for the identification. The first set consists of 2000 samples around the nominal set-point, whereas the second one consists of 961 equidistant steady-state data points in the range $q_1 \in [0, 30]$ and $q_3 \in [0, 30]$. Also, a dynamic test set with 2000 samples in the range $q_1 \in [0, 30]$ and $q_3 \in [0, 30]$ is also generated for model evaluation and model selection. All the data sets are scaled by the nominal process input ($\boldsymbol{u}_{\mathrm{norm}} = [16.6,\ 15.6]^{\mathrm{T}}$) and process output ($\boldsymbol{y}_{\mathrm{norm}} = [7.0255,\ 14.009]^{\mathrm{T}}$): $\boldsymbol{u} = \boldsymbol{u} - \boldsymbol{u}_{\mathrm{norm}}$, $\boldsymbol{y} = \boldsymbol{y} - \boldsymbol{y}_{\mathrm{norm}}$.

A five-order linear model ($n_x = 5$) is identified by the 'n4sid' algorithm[44], whose order has been proven to be sufficient to describe the dynamic of the neutralization reaction [25]. Then, to identify the PWL function, the steady-state intermediate variable vector $\boldsymbol{v}$ is computed as: $\boldsymbol{v} = [v_1, v_2]^T = \boldsymbol{K}_{\mathrm{lin}}^{-1} \times \boldsymbol{y}$.

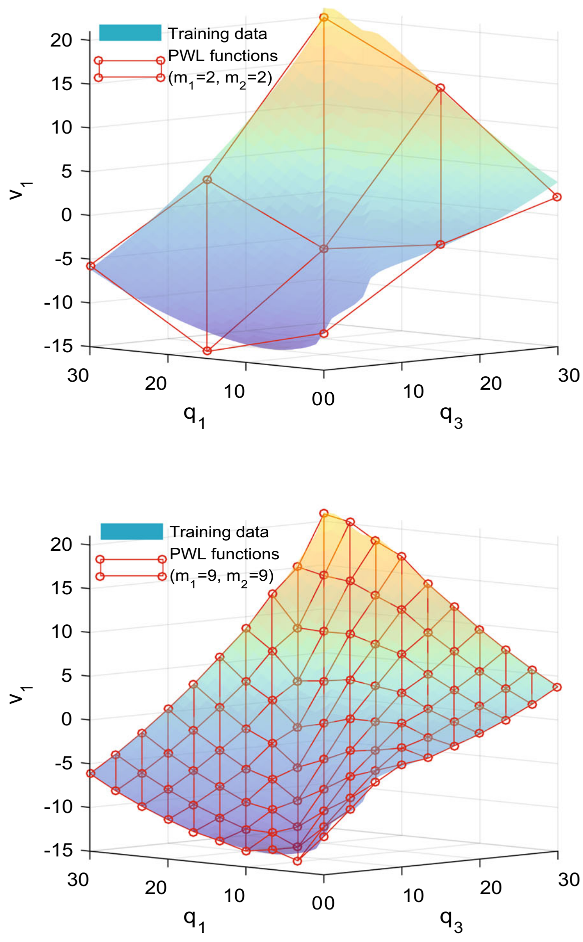
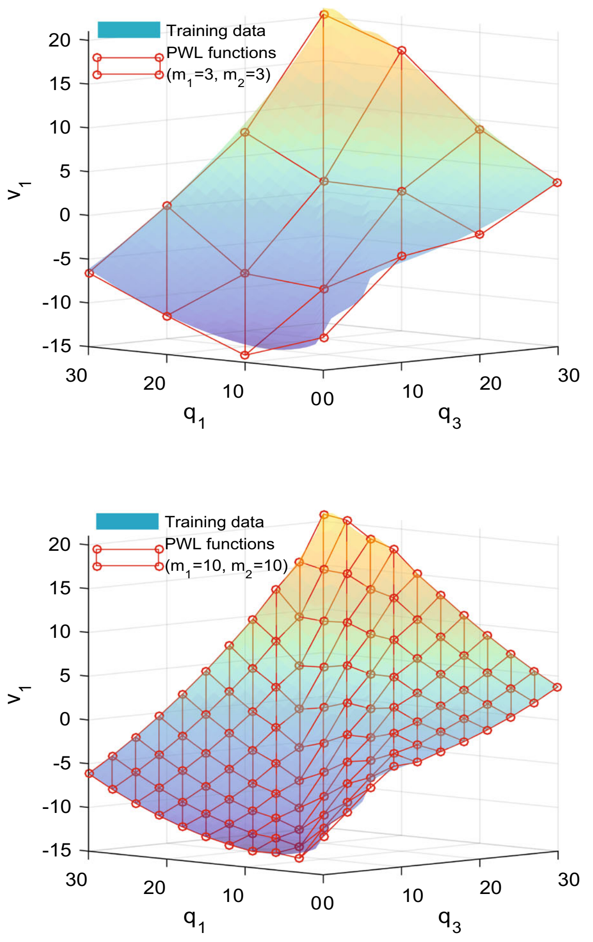


Fig. 10 Modeling of the pH reactor: the training steady-state data v_1 versus the obtained outputs of the PWL functions with different numbers of divisions



In this case, the influence of different numbers of divisions m_1 (associated with the q_1 variable) and m_2 (associated with the q_3 variable) on the overall accuracy of the final PWL Hammerstein model is studied. For simplicity, m_1 and m_2 are bundled together and are represented as $m_1 \times m_2$. The same values of m_1 and m_2 are always used. Moreover, the same configuration of $m_1 \times m_2$ grid is used for the two submodels, i.e., for the variables v_1 and v_2. Total nine kinds of PWL functions are identified for v_1 and v_2, whose number of divisions $(m_1 \times m_2)$ are $2 \times 2, 3 \times 3, 4 \times 4, \ldots 10 \times 10$, respectively. Figures 10 and 11 depict the steady-state data set versus the outputs of the PWL functions.

Finally, the PWL Hammerstein models are formed by combining the PWL functions and the linear model. The mean squared errors (MSE_{test}) for the test data set are defined to evaluate the Hammerstein models

$$\text{MSE}_{\text{test}} = \frac{1}{n_{\text{test}}} \sum_{k=1}^{n_{\text{test}}} \times \left[(\text{pH}_k - \text{pH}_k^{\text{mod}})^2 + (h_k - h_k^{\text{mod}})^2 \right] \tag{31}$$

where pH_k and h_k denote the real process outputs, pH_k^{mod} and h_k^{mod} are the outputs of the models. Table 5 shows test errors for the linear and PWL Hammerstein models with diffident numbers of divisions. Figure 12 depicts the outputs of the PWL Hammerstein models with different numbers of divisions and the outputs of the linear model. Apparently, the PWL Hammerstein models can approximate the pH reactor more precisely than the linear one. Considering the models with 2×2 and 3×3 divisions have relatively low model com-

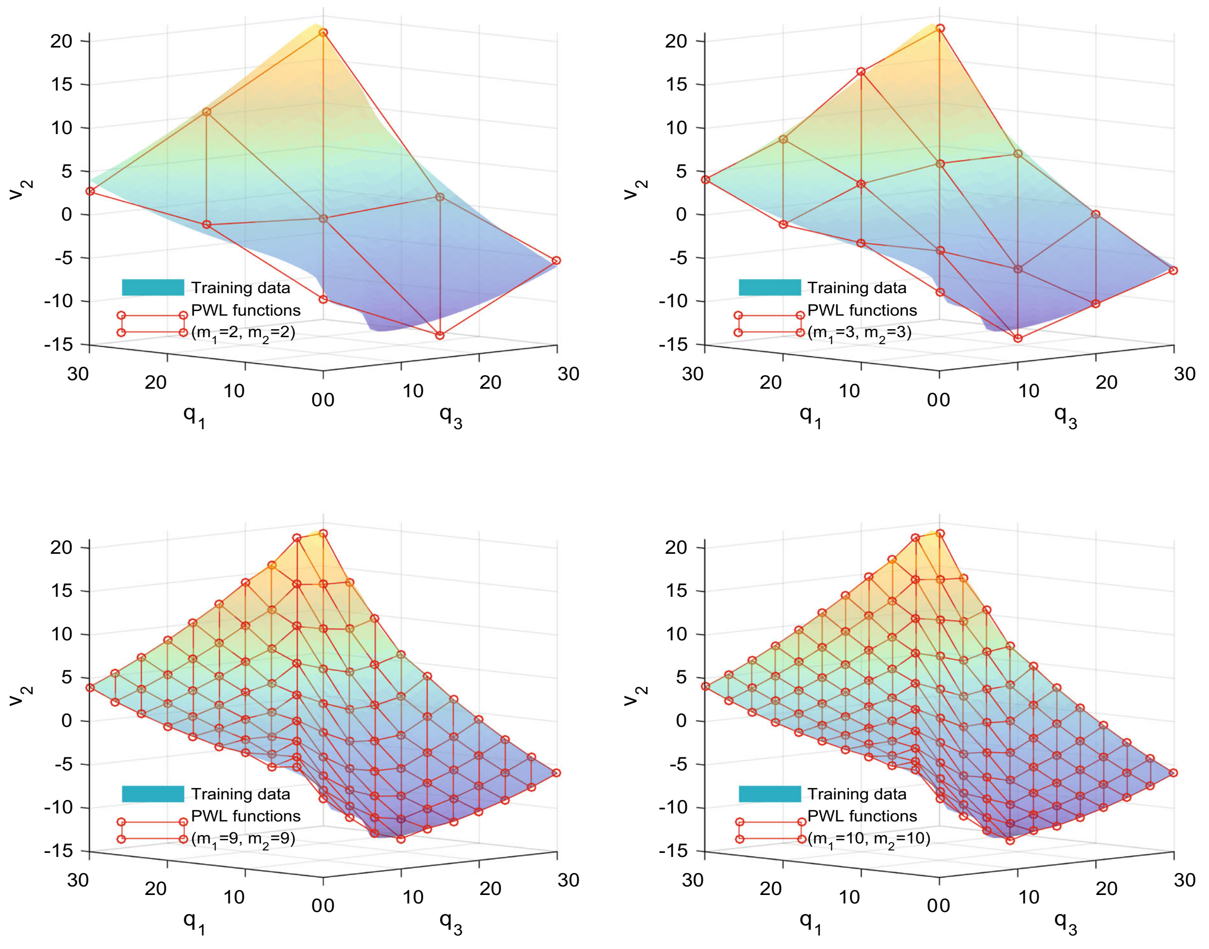


Fig. 11 Modeling of the pH reactor: the training steady-state data v_2 versus the obtained outputs of the PWL functions with different numbers of divisions

Table 5 Comparison of linear and PWL Hammerstein models of the pH reactor

Model	$m_1 \times m_2$	MSE_{test}
Linear	–	9.0496
PWL Hammerstein	2×2	3.0790
PWL Hammerstein	3×3	2.2955
PWL Hammerstein	4×4	2.2180
PWL Hammerstein	5×5	2.2062
PWL Hammerstein	6×6	2.1801
PWL Hammerstein	7×7	2.1344
PWL Hammerstein	8×8	2.0712
PWL Hammerstein	9×9	2.0590
PWL Hammerstein	10×10	2.0741

plexity and the models with 8×8 and 9×9 divisions have high accuracy, they are selected to form MPC-ML controllers.

4.2.2 Predictive control of the pH reactor

In this part, the PWL Hammerstein models with 2×2, 3×3, 8×8 and 9×9 divisions are used to form the MPC-ML controllers. The MPC-NL controllers are based on the PWL Hammerstein model with 9×9 divisions as this model has the highest modeling precision among the identified models. All MPC controllers use the same parameters: Hp =20, Hc = 10, $\boldsymbol{w}_y = \begin{bmatrix} 1 & 0 \\ 0 & 1000 \end{bmatrix}$, $\boldsymbol{w}_{\Delta u} = 0.5\boldsymbol{I}_{2\times2}$, $TV_u = 1$, and $t_{\max} = 3$. Constraints on the manipulated variables are: 10 mL/s $\leq q_1 \leq$ 20 mL/s and 10 mL/s $\leq q_3 \leq$ 20 mL/s.

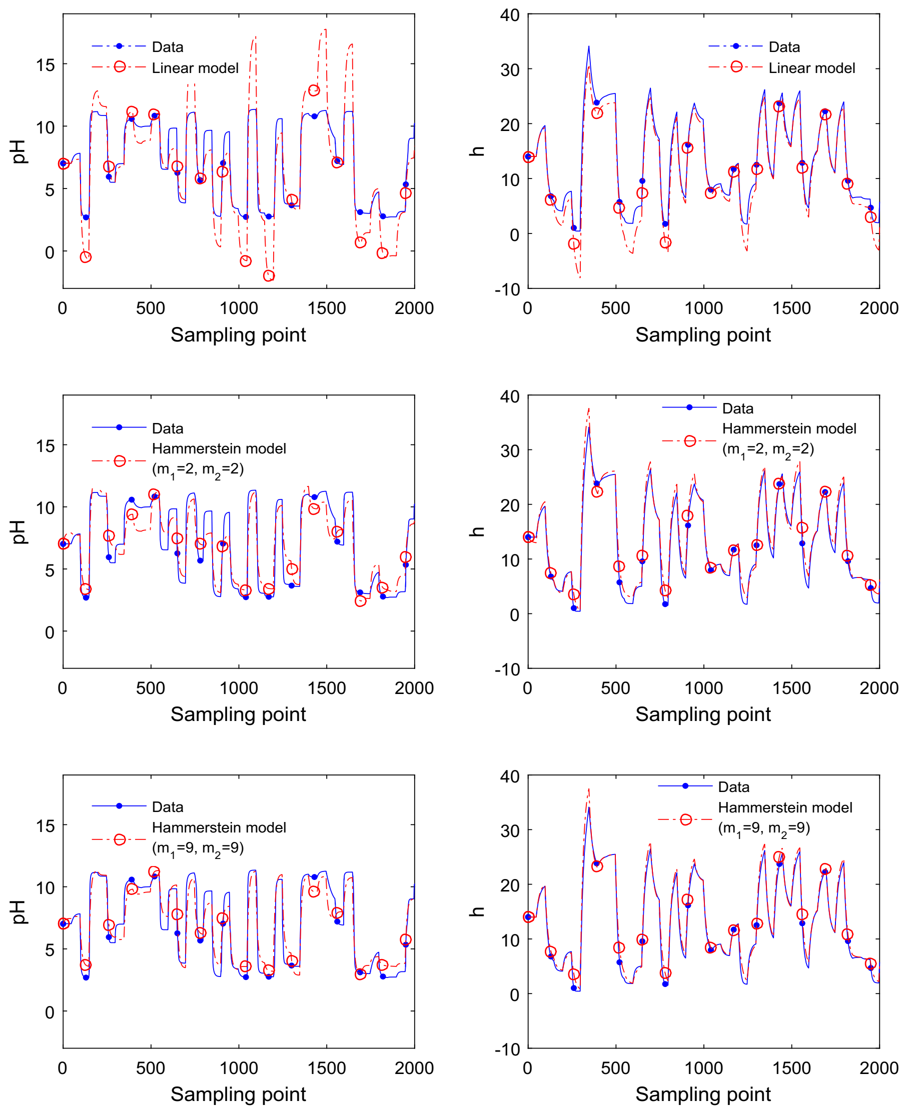


Fig. 12 Modeling of the pH reactor: the linear model versus the Hammerstein models with different numbers of divisions

Figures 13 and 14 show the closed-loop responses of the pH reactor for a predefined trajectory of set-point, with input and output disturbance models, respectively. To compare the MPC algorithms, the following integral square error (ISE) criterion is defined

$$\text{ISE} = \sum_{k=1}^{125} \left[(\text{pH}_k^r - \text{pH}_k)^2 + (h_k^r - h_k)^2 \right] \tag{32}$$

where pH_k^r and h_k^r denote the set-points. Table 3 compares the MPC controllers in terms of the ISE indicator and the scaled computation time (in relation to the most computationally demanding MPC-NL scheme with the input disturbance model, whose computation time is 30.46s).

The simulation results show that, in terms of the ISE indicator, the controllers based on the output disturbance model perform better than the controllers based on the input disturbance model. This means the output disturbance model is more appropriate and should be adopted for the pH neutralization process.

Compared with the linear controllers, the MPC-ML controllers with 2×2 and 3×3 divisions give worse control performance. That is because the corresponding PWL Hammerstein models are not sufficiently precise, and thus, the resulting MPC-ML controllers frequently switch model static gains between input subregions and

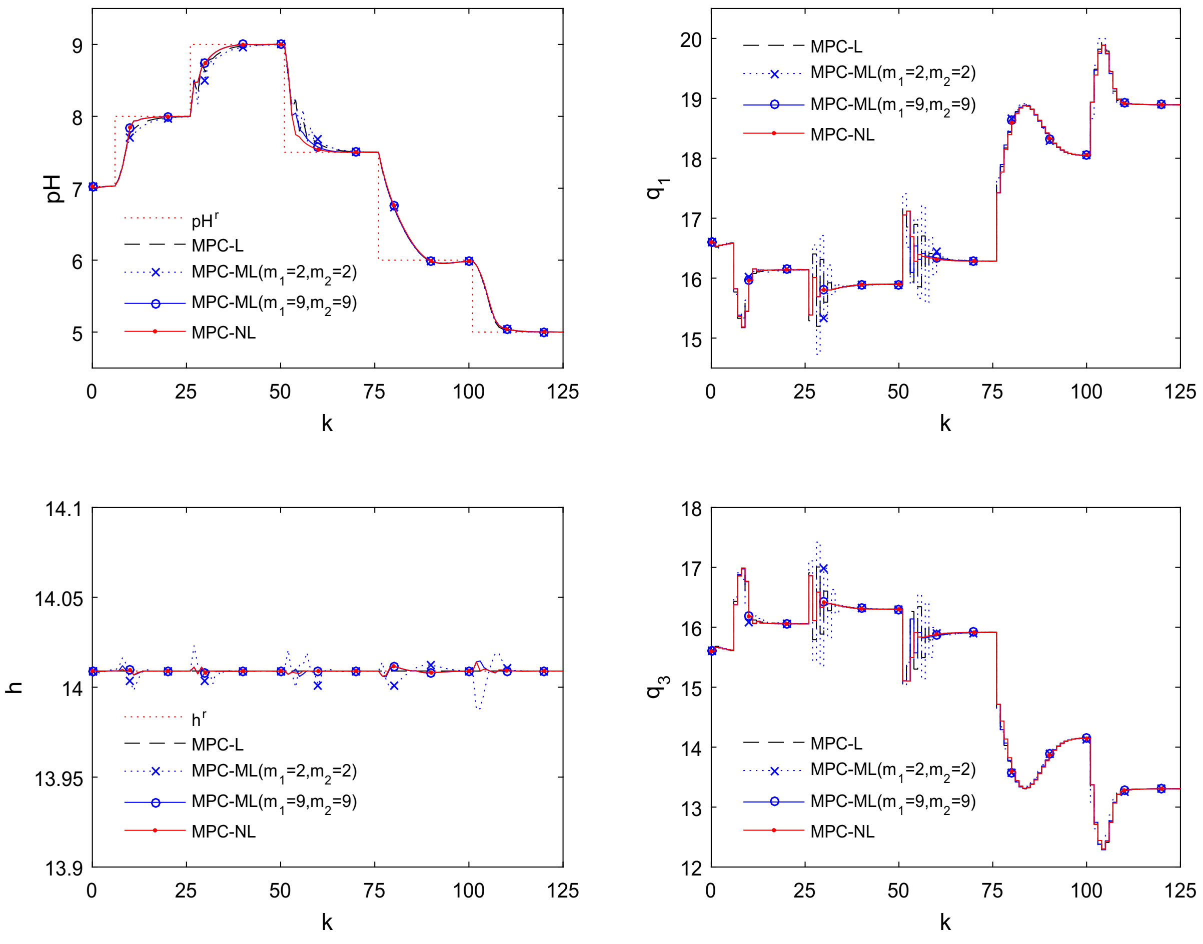


Fig. 13 Closed-loop responses of pH reactor with the input disturbance model

lead to output chatting (see Fig. 13) and large oscillations (see Fig. 14). Nevertheless, the proposed MPC-ML controllers wtih 8×8 and 9×9 divisions show much better control performance than the MPC-L controllers, which illustrates the necessity of using PWL Hammerstein models.

With the output disturbance model, the proposed MPC-ML controller with 9×9 divisions and the MPC-NPLPT controller give similar control performance, both of which perform just slightly poorer than the MPC-NL controller. However, the computation time of MPC-ML is significantly less than that of MPC-NPLPT and MPC-NL. All things considered, the MPC-ML controller based the output disturbance model and 9×9 divisions is recommended (Table 6).

5 Conclusion

In this study, a computationally efficient nonlinear MPC algorithm (MPC-ML) is developed for PWL Hammerstein models. By using the simplicial partition method, PWL functions are represented in the canonical style. Control laws are efficiently calculated via multistep linearization of the predicted output trajectory. A three-step procedure is designed to linearize PWL functions, where the derivatives of PWL functions are used in a simple look-up table manner. The most attractive advantage of this algorithm is that it can achieve a good balance between control accuracy and computational burden. Besides, it can integrate various disturbance models, which is likely to contribute to better control performance than the existing MPC-NPLPT algorithm. Inversion of input nonlinearity is not

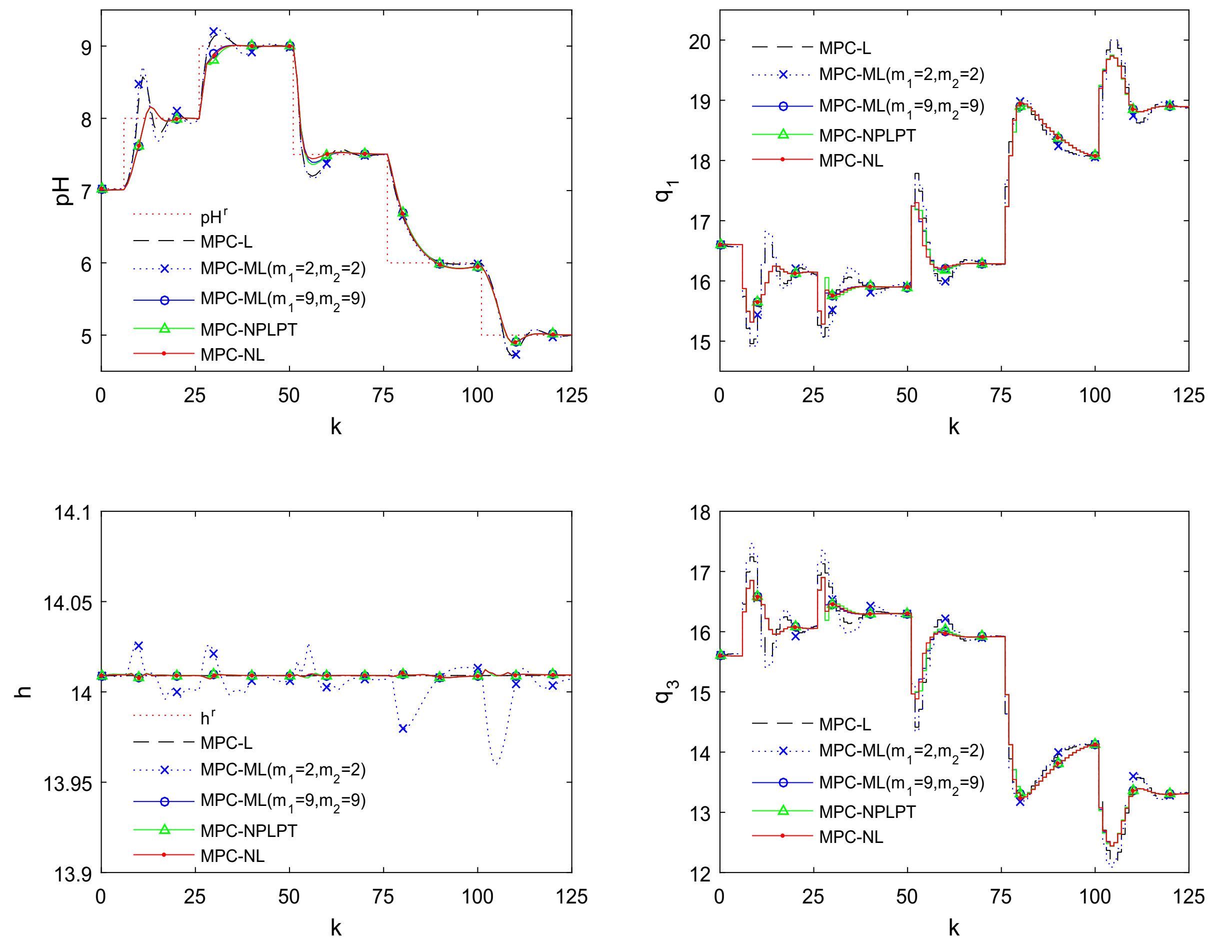


Fig. 14 Closed-loop responses of pH reactor with the output disturbance model

Table 6 Control accuracy and computational burden for the pH reactor

Algorithm	$m_1 \times m_2$	Disturbance	ISE	Scaled time (%)
MPC-L	–	Input	19.97	3.13
MPC-ML	2×2	Input	20.53	5.91
MPC-ML	3×3	Input	20.65	5.86
MPC-ML	8×8	Input	19.46	5.72
MPC-ML	9×9	Input	19.35	5.65
MPC-NL	9×9	Input	19.13	100.00
MPC-L	–	Output	18.33	3.17
MPC-ML	2×2	Output	18.91	5.67
MPC-ML	3×3	Output	18.87	5.64
MPC-ML	8×8	Output	18.27	5.57
MPC-ML	9×9	Output	18.21	5.59
MPC-NPLPT	–	Output	18.27	35.59
MPC-NL	9×9	Output	18.02	99.53

required by MPC-ML, and thus, it can directly integrate input constraints without any transformation.

The advantages of the proposed algorithm are demonstrated by two benchmark nonlinear processes: the CSTR and the pH reactor are considered. These two nonlinear processes can be precisely approximated by the PWL Hammerstein model with 5 divisions and the model with 9×9 divisions, respectively. In both the two cases, the proposed MPC-ML algorithm gives almost same control accuracy as in the MPC-NL algorithm with nonlinear optimization, whereas its computational time just accounts for around 5% of that consumed by the MPC-NL algorithm. All things considered, good control accuracy and computational efficiency are two main advantages of the described MPC-ML algorithm.

References

1. Maciejowski, J.M.: Predictive Control: with Constraints. Pearson Education, London (2002)
2. Darby, M.L., Nikolaou, M.: MPC: current practice and challenges. Control Eng. Pract. **20**, 328–342 (2012)
3. Grüne, L., Pannek, J.: Nonlinear Model Predictive Control, pp. 43–66. Springer, Berlin (2011)
4. Xu, Z., Zhao, J., Qian, J., Zhu, Y.: Nonlinear MPC using an identified LPV model. Ind. Eng. Chem. Res. **48**, 3043–3051 (2009)
5. Ławryńczuk, M.: Computationally Efficient Model Predictive Control Algorithms: A Neural Network Approach. Studies in Systems, Decision and Control. Springer, Cham (2014)
6. Zhang, R., Wang, S.: Support vector machine based predictive functional control design for output temperature of coking furnace. J. Process Control **18**, 439–448 (2008)
7. Fruzzetti, K., Palazoğlu, A., McDonald, K.: Nolinear model predictive control using Hammerstein models. J. Process Control **7**, 31–41 (1997)
8. Giri, F., Bai, E.W.: Block-Oriented Nonlinear System Identification, vol. 1. Springer, London (2010)
9. Patwardhan, R.S., Lakshminarayanan, S., Shah, S.L.: Constrained nonlinear MPC using Hammerstein and Wiener models: PLS framework. AIChE J. **44**, 1611–1622 (1998)
10. Chan, K.H., Bao, J.: Model predictive control of Hammerstein systems with multivariable nonlinearities. Ind. Eng. Chem. Res. **46**, 168–180 (2007)
11. Du, J., Song, C., Li, P.: Multilinear model control of Hammerstein-like systems based on an included angle dividing method and the MLD-MPC strategy. Ind. Eng. Chem. Res. **48**, 3934–3943 (2009)
12. Menold, P., Allgöwer, F., Pearson, R.: Nonlinear structure identification of chemical processes. Comput. Chem. Eng. **21**, S137–S142 (1997)
13. Huo, H., Zhu, X., Hu, W., Tu, H., Li, J., Yang, J.: Nonlinear model predictive control of SOFC based on a Hammerstein model. J. Power Sour. **185**, 338–344 (2008)
14. Jalaleddini, K., Kearney, R.E.: Subspace identification of SISO Hammerstein systems: application to stretch reflex identification. IEEE Trans. Biomed. Eng. **60**, 2725–2734 (2013)
15. Zhang, Q., Wang, Q., Li, G.: Nonlinear modeling and predictive functional control of Hammerstein system with application to the turntable servo system. Mech. Syst. Signal Process. **72**, 383–394 (2016)
16. Ławryńczuk, M.: Practical nonlinear predictive control algorithms for neural Wiener models. J. Process Control **23**, 696–714 (2013)
17. Ławryńczuk, M.: Nonlinear predictive control based on least squares support vector machines Hammerstein models. In: International Conference on Adaptive and Natural Computing Algorithms, pp. 246–255. Springer (2013)
18. Julian, P., Desages, A., Agamennoni, O.: High-level canonical piecewise linear representation using a simplicial partition. IEEE Trans. Circuits Syst. I: Fundam. Theory Appl. **46**, 463–480 (1999)
19. Julian, P.: A high level canonical piecewise linear representation: theory and applications. Ph. D. Thesis: Universidad Nacional del Sur, Bahia Blanca, Argentina (1999)
20. Julian, P.: PWL Matlab Toolbox. http://uns.academia.edu/PedroJulian/Matlab-PWL-Toolbox (2000)
21. Sentoni, G., Agamennoni, O., Desages, A., Romagnoli, J.: Approximate models for nonlinear process control. AIChE J. **42**, 2240–2250 (1996)
22. Bloemen, H., Van den Boom, T., Verbruggen, H.: Model-based predictive control for Hammerstein systems. In: Proceedings of the 39th IEEE Conference on Decision and Control, vol. 5, pp. 4963–4968. IEEE (2000)
23. Bloemen, H., Van Den Boom, T., Verbruggen, H.: Model-based predictive control for Hammerstein-Wiener systems. Int. J. Control **74**, 482–495 (2001)
24. Wang, H., Zhao, J., Xu, Z., Shao, Z.: Model predictive control for Hammerstein systems with unknown input nonlinearities. Ind. Eng. Chem. Res. **53**, 7714–7722 (2014)
25. Ławryńczuk, M.: Suboptimal nonlinear predictive control based on multivariable neural Hammerstein models. Appl. Intell. **32**, 173–192 (2010)
26. Chua, L.O., Kang, S.M.: Section-wise piecewise-linear functions: canonical representation, properties, and applications. Proc. IEEE **65**, 915–929 (1977)
27. Oblak, S., Škrjanc, I.: Continuous-time Wiener-model predictive control of a pH process based on a PWL approximation. Chem. Eng. Sci. **65**, 1720–1728 (2010)
28. Yu, F., Mao, Z., Jia, M.: Recursive identification for Hammerstein-Wiener systems with dead-zone input nonlinearity. J. Process Control **23**, 1108–1115 (2013)
29. Borrelli, F., Bemporad, A., Morari, M.: Predictive Control for Linear and Hybrid Systems. Cambridge University Press, Cambridge (2013)
30. Gonzalez, A., Adam, E., Marchetti, J.: Conditions for offset elimination in state space receding horizon controllers: a tutorial analysis. Chem. Eng. Process. **47**, 2184–2194 (2008)
31. Maeder, U., Borrelli, F., Morari, M.: Linear offset-free model predictive control. Automatica **45**, 2214–2222 (2009)
32. Muske, K.R., Badgwell, T.A.: Disturbance modeling for offset-free linear model predictive control. J. Process Control **12**, 617–632 (2002)
33. Pannocchia, G., Rawlings, J.B.: Disturbance models for offset-free model-predictive control. AIChE J. **49**, 426–437 (2003)

34. Rawlings, J.B., Mayne, D.Q.: Model Predictive Control: Theory and Design. Nob Hill Publishing, Madison (2009)
35. Wang, L.: Model Predictive Control System Design and Implementation Using MATLAB®. Springer, Berlin (2009)
36. Zhang, R., Xue, A., Wang, S., Ren, Z.: An improved model predictive control approach based on extended non-minimal state space formulation. J. Process Control **21**, 1183–1192 (2011)
37. Zhang, J.: Improved decoupled nonminimal state space model based PID for multivariable processes. Ind. Eng. Chem. Res. **54**, 1640–1645 (2015)
38. Morari, M., Maeder, U.: Nonlinear offset-free model predictive control. Automatica **48**, 2059–2067 (2012)
39. Rao, C.V., Rawlings, J.B., Lee, J.H.: Constrained linear state estimation–a moving horizon approach. Automatica **37**, 1619–1628 (2001)
40. Yue, Y., Li, H., Shao, W., Wu, B.: Nonlinear model predictive control based on Hammerstein piecewise linear models. In: Proceedings of the 32nd Chinese Control Conference (2013)
41. Ławryńczuk, M.: Nonlinear predictive control for Hammerstein-Wiener systems. ISA Trans. **55**, 49–62 (2015)
42. Ławryńczuk, M.: Nonlinear predictive control of dynamic systems represented by Wiener-Hammerstein models. Nonlinear Dyn. **86**, 1193–1214 (2016)
43. Verhaegen, M., Westwick, D.: Identifying MIMO Hammerstein systems in the context of subspace model identification methods. Int. J. Control **63**, 331–349 (1996)
44. Van Overschee, P., De Moor, B.: N4sid: subspace algorithms for the identification of combined deterministic-stochastic systems. Automatica **30**, 75–93 (1994)
45. Cao, X., Ayalew, B.: Control-oriented MIMO modeling of laser-aided powder deposition processes. In: American Control Conference (ACC), pp. 3637–3642. IEEE (2015)
46. Marusak, P.M.: Advantages of an easy to design fuzzy predictive algorithm in control systems of nonlinear chemical reactors. Appl. Soft Comput. **9**, 1111–1125 (2009)