I&EC research
Industrial & Engineering Chemistry Research

Article

Cite This: *Ind. Eng. Chem. Res.* 2017, 56, 14101−14114

pubs.acs.org/IECR

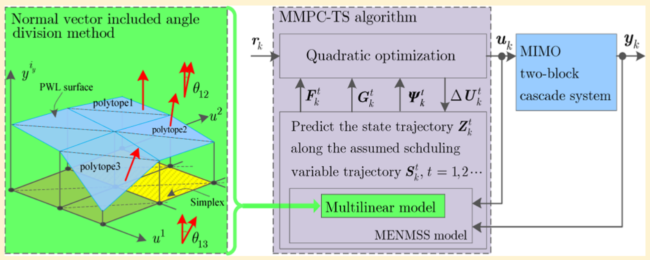
Multilinear Model Decomposition and Predictive Control of MIMO Two-Block Cascade Systems

Jian Zhang,[*,†] Kwai Sang Chin,[†] and Maciej Ławryńczuk[‡]

[†]Department of System Engineering and Engineering Management, City University of Hong Kong, 83 Tatchee Avenue, Kowloon, Hong Kong

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

ABSTRACT: This paper presents a new multilinear model decomposition method for multiple-input multiple-output (MIMO) two-block cascade systems and a model predictive control (MPC) algorithm for the resulting representation. First, a normal vector included angle division method is developed to decompose the operating space and determine the minimum linear model bank through evaluating the nonlinearity of the steady-state I/O surfaces. For a prescribed angle threshold, the minimum linear model bank can be constructed to approximate the original two-block cascade system sufficiently closely. Next, a multilinear MPC algorithm is designed with the proposed trajectory scheduling technique, which can reduce output oscillations caused by hard switching and avoid the difficulty of calculating/tuning complex weighting functions/parameters used in soft switching. A benchmark chemical reactor process is studied to illustrate the effectiveness and advantages of the proposed decomposition method and the predictive control algorithm.



1. INTRODUCTION

The multilinear model control approach is an effective control technique for nonlinear processes with wide operating ranges and large set-point changes. Its key concept is to first approximate a nonlinear process as a combination of multiple linear models and then design the system controller based on classical control techniques. Various multilinear model control algorithms have been proposed and applied in practical systems.[1,2]

The most crucial problem of multilinear model control is how to decompose a nonlinear system, namely, *how many and which linear models are required to span the expected operating space of a nonlinear system.*[2−4] This problem has not been completely solved and is still under study. Determination of the number and the positions of linear submodels is usually performed experimentally, but the trial and error approach is really time-consuming and costly. In ref 5, Galán et al. first proposed using the gap metric concept to compare the candidate models and reduce the number of linear submodels in a given model bank. Tan et al.[6] extended the method in Galán et al.,[5] where operating point selection is integrated with local controller design via the loop-shaping H_∞ approach. However, the operating points were selected from an existing point set, and the selection requires advance knowledge of the achievable closed-loop performance. Based on gap metric, Du et al.[4] first designed the operating range division method for a nonlinear system with one scheduling variable, which does not need the assumption of existing model banks or operating point sets. Then, Du et al.[7] generalized the method in ref 4 to accommodate nonlinear systems with multiple scheduling variables, together with a gridding algorithm for reducing gridding points and decreasing computational burden. To accurately measure the distance between candidate linear models in all ranges of frequencies, Shaghaghi et al.[8] studied the H-gap metric based division method.

All the aforementioned methods divide the nonlinear system by comparing local linear models. Such methods depend on the system dynamic information around operating points and are just effective in the case where a set of local linear models is available before dividing, such as the cases in refs 5 and 6. Once local linear models are not available, system nonlinear models or a large number of local linear models would have to be identified from measured data, such as the cases in refs 4, 7, and 8. Because identifying an accurate nonlinear model or a large number of local linear models is generally time-consuming and costly, the gap metric based division methods are strictly restricted by the requirement of system dynamic information. Nevertheless, for a general nonlinear system, there seems to be no better alternative than the gap metric based methods.

In contrast, for the Hammerstein system (a nonlinear steady-state block followed by a linear dynamic one) and the Wiener system (a linear dynamic block followed by a nonlinear steady-state one), their static I/O mappings (curves or surfaces) can sufficiently represent the nonlinearity and can be used for

Received: August 7, 2017
Revised: November 7, 2017
Accepted: November 8, 2017
Published: November 8, 2017

multilinear model decomposition. As the two systems both consist of a nonlinear static block and a linear dynamic subsystem, they are called a "*two-block cascade system*" in this article. In ref 9, Du et al. propose an included angle division method for single-input-single-output (SISO) two-block cascade systems. The difference between two slope angles at two operating points in the static I/O curve is employed to represent the change of system linearity. Thus, this method just requires the static I/O curve instead of entire system dynamic information. This method is simple and intuitive, but it is not applicable for MIMO systems because, at each operating point of a MIMO system, there are an infinite number of slopes in different directions on its I/O surface. On the other hand, MIMO two-block cascade systems with more than one scheduling variable exist widely in industry, such as the MIMO continuous stirred tank reactor,[10,11] the polymerization reactor,[10] the binary distillation column,[12] the laser-aided powder deposition process,[13] and so on. Therefore, a low-cost but effective multilinear model division method is urgently needed for MIMO two-block cascade systems.

Once the multilinear model is obtained, the local linear controllers can be initially designed by using many types of control methods (e.g., MPC, PID, and LQR). MPC is recommended here because of its natural ability to deal with MIMO systems and constraints.[14] The main problem in controller design is *how to integrate these local MPC controllers as a global one*. The simplest method is the hard switching method,[2] in which only one local controller is selected to calculate the control law and local controllers are switched between different operating subregions. Hard switching of controllers usually leads to unacceptable output oscillations, e.g., the unexpected impulsive phenomena. To reduce output oscillations, many soft switching methods have been proposed based on different weighting functions, such as the Gaussian function based method,[15] the Bayesian weighting function based method,[16] the gap metric based method,[17,18] and so on. However, designing a proper weighting function is generally not easy, because usually the weighting parameters can only be determined by engineering experience and calculation of gap metric requires complete system dynamic information. For these reasons, a simple but smooth switching method is expected for local MPC controller combination.

This study focuses on the multilinear model decomposition and control of MIMO two-block cascade systems. The main contributions of this paper are summarized as follows:

1. The proposed normal vector included angle division method. This method extends the idea in ref 9 to MIMO systems by using the normal vector included angle concept. It initially partitions the system I/O surface into a certain number of polytopes and then employs the included angles between the normal vectors to these polytopes to evaluate system nonlinearity and decompose the system. It is usually more practical than the gap metric based methods,[4−8] because only static I/O mappings instead of system dynamic information are required for the system decomposition.
2. The designed multilinear MPC algorithm with trajectory scheduling (MMPC-TS). At each sampling instant, by using a scheduling variable trajectory, the multilinear predictive model is transformed to be a certain local linear model within the prediction horizon. The proposed trajectory scheduling technique enables MMPC-TS to reduce output oscillation in comparison with the control algorithms with hard switching, and meanwhile it avoids the complex weighting functions used in soft switching.

A benchmark MIMO continuous stirred tank reactor system is studied to illustrate the effectiveness of the proposed division method and the developed control algorithm.

The rest of this paper is organized as follows. Section 2 introduces the two-block cascade system and its multilinear model. Section 3 discusses the normal vector included angle division method. Section 4 details development of the nonlinear MPC based on the multilinear model. Section 5 gives the simulation results for the MIMO CSTR benchmark system. Section 6 concludes the paper.

2. MIMO TWO-BLOCK CASCADE SYSTEM AND ITS MULTILINEAR MODEL

In this article, the two-block cascade system is a general term for the Hammerstein system

$$\begin{cases} \boldsymbol{v}_k = f(\boldsymbol{u}_k) \\ \boldsymbol{y}_k = -\sum_{i=1}^{n} \bar{\boldsymbol{F}}_i \boldsymbol{y}_{k-i} + \sum_{i=1}^{n} \bar{\boldsymbol{H}}_i \boldsymbol{v}_{k-i} \end{cases} \tag{1}$$

and the Wiener system

$$\begin{cases} \boldsymbol{w}_k = -\sum_{i=1}^{n} \bar{\boldsymbol{F}}_i \boldsymbol{w}_{k-i} + \sum_{i=1}^{n} \bar{\boldsymbol{H}}_i \boldsymbol{u}_{k-i} \\ \boldsymbol{y}_k = g(\boldsymbol{w}_k) \end{cases} \tag{2}$$

where $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ is the system input vector (the vector of manipulated variables), $\boldsymbol{v}_k \in \mathbb{R}^{n_v}$ and $\boldsymbol{w}_k \in \mathbb{R}^{n_w}$ are intermediate variable vectors, $\boldsymbol{y}_k \in \mathbb{R}^{n_y}$ is the system output vector (the vector of controlled variables), k is the sample time, $\bar{\boldsymbol{F}}_i$ and $\bar{\boldsymbol{H}}_i$ are the linear coefficient matrices, $f(\cdot)\colon \mathbb{R}^{n_u} \to \mathbb{R}^{n_v}$ and $g(\cdot)\colon \mathbb{R}^{n_w} \to \mathbb{R}^{n_y}$ are static nonlinear functions.

Note the static nonlinear functions are not necessarily invertible for this study. The only assumption on the nonlinear parts is the functions $f(\cdot)$ and $g(\cdot)$ are Lipschitz-continuous.[19] This assumption is not restrictive because the static description of numerous technological processes[10−13] satisfies the Lipschitz-continuous condition.

The multilinear model of a two-block system is a collection of multiple local linear submodels, with designated operating subregions and operating points scheduled by the scheduling variables. Typically, the scheduling variables can be system inputs, outputs, states, and disturbances. Let $\boldsymbol{s}$ denote the vector of scheduling variables. The variation range of $\boldsymbol{s}$ is the operating space of the system, denoted as Φ. Suppose Φ is divided into N_m subregions Φ_i ($i = 1, ..., N_m$). Φ_i should satisfy $\Phi_i \subseteq \Phi$, $\Phi_1 \cup ... \cup \Phi_{N_m} = \Phi$ and $\Phi_i \cap \Phi_j = \emptyset$ for $i \neq j$.[7] For each subregion Φ_i (i = 1, ..., N_m), an operating point is assigned, denoted as OP$_i$ ($\boldsymbol{u}_o^i$,$\boldsymbol{y}_o^i$). To facilitate the following operating space division, the system inputs are selected as the scheduling variables, i.e., $\boldsymbol{s} = \boldsymbol{u}$, and thereby the operating space Φ becomes the input space. In the following, dividing the operating space is equivalent to dividing the input space.

In the ith subregion Φ_i, a local linear submodel can be acquired around OP$_i$ ($\boldsymbol{u}_o^i$,$\boldsymbol{y}_o^i$) to approximate the studied two-block cascade system, which is expressed by the following difference equation:

$$\delta y_k + F_1^i \delta y_{k-1} + ... + F_n^i \delta y_{k-n} = H_1^i \delta u_{k-1} + H_2^i \delta u_{k-2} + ... + H_n^i \delta u_{k-n} + \delta \varepsilon_k,\ s \in \Phi_i \qquad (3)$$

where $\delta u_k = u_k - u_o^i$, $\delta y_k = y_k - y_o^i$, F_1^i, ..., F_n^i, and H_1^i, ..., H_n^i are model coefficients. The error term $\delta \varepsilon_k = \varepsilon_k - \varepsilon_o^i$, where $\delta \varepsilon_k$ denotes the general modeling error including model mismatch, external disturbances, etc., and ε_o^i is the steady-state modeling error at the operating point OP$_i$. Equation 3 is the multilinear model of a two-block cascade system, which is also called the linear model bank because it contains N_m linear submodels.

The two-block cascade system has the important property that its nonlinearities are *"primarily"* static whereas the dynamic characteristics are *"basically"* linear. That means only the static system gain changes with operating points whereas the system dynamics almost remain the same in the full operating space. Thus, for a MIMO two-block cascade system, its static I/O surfaces can provide enough information for operating space division and multilinear model construction.

3. NORMAL VECTOR INCLUDED ANGLE DIVISION METHOD

In this section, a division method based on system I/O surfaces is developed for MIMO tow-block cascade systems. The steady-state I/O mappings are always assumed to be already *available*. In practice, they can be obtained by system identification or by first principle modeling. For most systems, such as chemical processes[10−13] and large-scale wind tunnels,[20] the static I/O mappings are generally more readily available and much easier to identify than detailed dynamic models, especially for the systems with available controllers (even if the controllers may not have perfect performance, e.g., simple PID controllers). Therefore, this method is more practical and lower-cost than dynamic information based division methods.[4−8]

3.1. Normal vector included angle concept. To divide the operating space, some metric must be found to evaluate the nonlinearity of the system I/O surfaces. Based on the following two properties, the normal vector included angle is proposed to act as the metric:

1. *Any Lipschitz-continuous functions defined on a compact domain can be approximated arbitrarily closely by a PieceWise Linear (PWL) function based on the so-called simplicial partition.*[21,22] This means a I/O surface characterized by a Lipschitz-continuous nonlinear function can be sufficiently approximated by a finite number of polytopes, as shown in Figure 1. Each polytope corresponds to a simplex region in the input space.

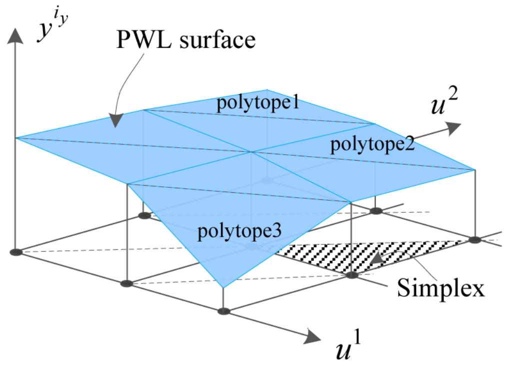


Figure 1. Piecewise linear function approximation.

Thus, we can assess the nonlinearity of the I/O surface by indirectly analyzing the nonlinearity between these polytopes, and we finally divide the input space (i.e., the operating space).

2. *The normal vector included angle of two hyperplanes can effectively assess the nonlinearity between the two hyperplanes.* Suppose the system I/O surface has been well approximated by some polytopes, each of which is contained within a hyperplane of dimensionality n_u and corresponds to a unique input region (a simplex). If the normal vector included angle of two hyperplanes is sufficiently small (e.g., θ_{12} in Figure 2), we can believe there is not system nonlinearity and the studied system has a same static gain in the two input regions. Otherwise, if the included angle is very large (e.g., θ_{13} in Figure 2), we can believe the studied system has significant nonlinearity in the two input regions.

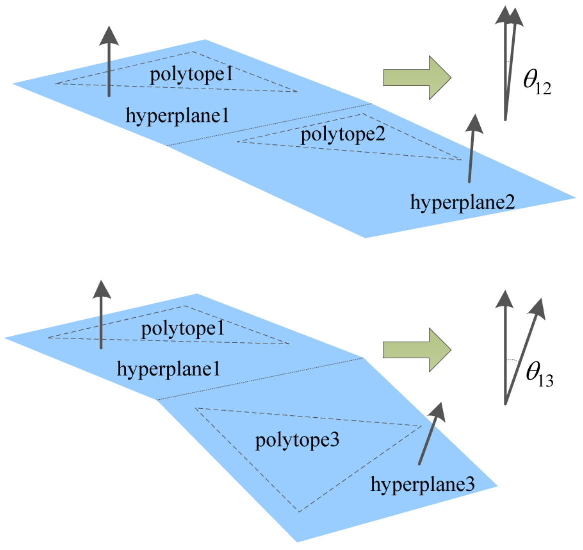


Figure 2. Normal vector included angles.

Calculation of the normal vector included angle consists of two steps. First, calculate the normal vector to a hyperplane. This can be achieved by using the hyperplane equation or by n_u linear independent points on this hyperplane. Because the n_u vertexes of a polytope, donated as pt_1, pt_2, ..., pt_{n_u+1}, are naturally linear independent, they are used to compute the corresponding normal vector $n = [n^1\ n^2 ...\ n^{n_u+1}]^T$, namely

$$\begin{cases} n \cdot (pt_i - pt_j) = 0,\ 1 \le i < j \le n_u + 1 \\ n^{n_u+1} = 1 \end{cases} \qquad (4)$$

In eq 4, $pt_i - pt_j$ $(1 \le i < j \le n_u + 1)$ are actually n_u noncollinear vectors on the hyperplane. The last equation "$n^{n_u+1} = 1$" indicates all the calculated normal vector is assigned to be the positive direction of the y^{i_y} $(1 \le i_y \le n_y)$ axis. The constant 1 can be replaced by other nonzero constants, which would not influence the final included angle.

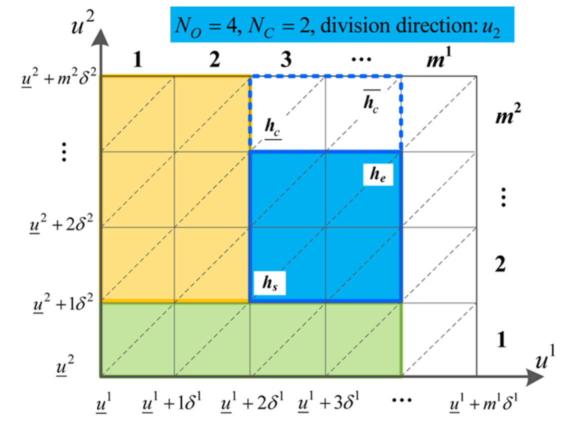
Once two normal vectors (e.g., n_i and n_j) are obtained, the included angle between them can be easily computed as

$$\theta(n_i, n_j) = \arccos\left(\frac{n_i \cdot n_j}{|n_i||n_j|}\right) \qquad (5)$$

where $|\cdot|$ represents the magnitude of the vector.

The range of included angle is $[0, \pi]$. If the included angle is very small (e.g., 0), the system has similar static gains and behaves similarly at the two regions; otherwise, the static gains and the system behaviors change significantly from one region to the other. In this study, the normal vector included angle is used to evaluate the difference of system behavior at the two operating regions and further to develop the division method.

3.2. Subprocedure1: simplicial partition of the input space. Figure 3 shows the division scheme. Prior to introducing the main procedure of the division method, two subprocedures are initially proposed according to the theoretical basis in section 3.1, based on which main procedure is developed in section 3.4.



$N_O = 4, N_C = 2$, division direction: u_2
u^2
u^1
1
2
3
…
m^1
m^2
…
2
1
$\underline{u}^2 + m^2\delta^2$
$\underline{u}^2 + 2\delta^2$
$\underline{u}^2 + 1\delta^2$
$\underline{u}^2$
$\underline{u}^1$
$\underline{u}^1 + 1\delta^1$
$\underline{u}^1 + 2\delta^1$
$\underline{u}^1 + 3\delta^1$
$\underline{u}^1 + m^1\delta^1$
$\underline{h}_c$
$\overline{h}_c$
h_e
h_s

Figure 3. Normal vector included angle based division of a operating space in $\mathbb{R}^2$.

The first subprocedure of the division method is to partition the input space. Consider the following input space

$$\boldsymbol{D} = \{\boldsymbol{u} \in \mathbb{R}^{n_u} | \underline{u}^{i_u} \le u^{i_u} \le \overline{u}^{i_u},\ i_u = 1, 2, ..., n_u\} \tag{6}$$

where $\underline{u}^{i_u}$ and $\overline{u}^{i_u}$ are the lower bound and the upper bound of u^{i_u}, respectively.

The simplicial partition of the input space consists of two steps.[21,22] First, by specifying the number of divisions (m^{i_u}) associated with the u^{i_u} axis (i.e., partitioning the interval $[\underline{u}^{i_u}, \overline{u}^{i_u}]$ into m^{i_u} subintervals), the domain $\boldsymbol{D}$ is evenly partitioned into $\prod_{i=1}^{n_u} m^{i_u}$ hypercubes. Each hypercube, denoted as $\boldsymbol{H_h}$, is indexed by a n_u-dimension vector $\boldsymbol{h} = [h^1\ h^2, ..., h^{n_u}]^{\mathrm{T}}$, where h^{i_u} $(1 \le i_u \le n_u)$ indicates the hypercube locates at the i_uth subinterval on the u^{i_u} axis. Accordingly, the grid step in the axis u^i is $\delta^{i_u} = (\overline{u}^{i_u} - \underline{u}^{i_u})/m^{i_u}$. Second, each hypercube is subdivided into $n_u!$ simplices. By using the two-step simplicial partition, the input space $\boldsymbol{D}$ is finally partitioned into $\prod_{i=1}^{n_u} m^{i_u} \times n_u!$ simplices. More details about the simplicial partition are found in refs 21 and 22. In this subprocedure, the number of divisions are designer-assigned parameters, which are written in the vector $\boldsymbol{m} = [m^1, m^2, ..., m^{n_u}]^{\mathrm{T}}$ for notational simplicity.

Based on the above simplicial partition, a unique continuous high dimensional surface composed of polytopes can be found to approximate the original I/O surface, as shown in Figure 1. Namely, the system I/O surface is approximated as a set of certain polytopes.

3.3. Subprocedure2: combining subregions. In the subprocedure1, the input space is initially partitioned into hypercube subregions and then is partitioned into different simplex subregions. The subprocedure2 will discuss how to systematically combine those simplices to form the final operating subregions.

To simplify the following subregion combination and facilitate the description of the final operating subregion, this study regards a hypercube subregion as the basic division unit and finally the entire input space is divided into several hypercuboid regions. The issue of combining two hypercube subregions is studied in subsection 3.3.1, based on which subsection 3.3.2 discusses the method to combine hypercuboid subregions.

3.3.1. Hypercube subregions combination. According to the simplicial partition in section 3.2, in the input space each hypercube $\boldsymbol{H_h}$ contains $n!$ simplices and each simplex corresponds to a polytope approximating the I/O surface. Thus, for a given I/O surface, we can calculate $n!$ normal vectors for a hypercube input subregion $\boldsymbol{H_h}$.

Consider two hypercubes $\boldsymbol{H}_{\boldsymbol{h}_i}$ and $\boldsymbol{H}_{\boldsymbol{h}_j}$ in the input space (where $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ are two index vectors). For the i_yth ($i_y = 1, ..., n_y$) I/O surface, two sets of normal vectors can be calculated for $\boldsymbol{H}_{\boldsymbol{h}_i}$ and $\boldsymbol{H}_{\boldsymbol{h}_j}$:

$$\mathbf{N}_{\boldsymbol{h}_i}^{i_y} = \{\boldsymbol{n}_{\boldsymbol{h}_i}^{i_y,1}, \boldsymbol{n}_{\boldsymbol{h}_i}^{i_y,2} ..., \boldsymbol{n}_{\boldsymbol{h}_i}^{i_y,n_u!}\},\ \mathbf{N}_{\boldsymbol{h}_j}^{i_y} = \{\boldsymbol{n}_{\boldsymbol{h}_j}^{i_y,1}, \boldsymbol{n}_{\boldsymbol{h}_i}^{i_y,2} ..., \boldsymbol{n}_{\boldsymbol{h}_j}^{i_y,n_u!}\} \tag{7}$$

These two sets of normal vectors contain the system static information, which is used to evaluate system nonlinearity between the two hypercube regions. Through pairwise comparison on the two sets of normal vectors, an included angle vector with $n_u! \times n_u!$ elements is formed as the follows:

$$\boldsymbol{\theta}_{\boldsymbol{h}_i,\boldsymbol{h}_j}^{i_y} = [\theta(\boldsymbol{n}_{\boldsymbol{h}_i}^{i_y,1}, \boldsymbol{n}_{\boldsymbol{h}_j}^{i_y,1}), \theta(\boldsymbol{n}_{\boldsymbol{h}_i}^{i_y,1}, \boldsymbol{n}_{\boldsymbol{h}_j}^{i_y,2}), ..., \theta(\boldsymbol{n}_{\boldsymbol{h}_i}^{i_y,n_u!}, \boldsymbol{n}_{\boldsymbol{h}_j}^{i_y,n_u!})]^{\mathrm{T}}_{(n_u!\times n_u!)\times 1} \tag{8}$$

The largest included angle in $\boldsymbol{\theta}_{\boldsymbol{h}_i,\boldsymbol{h}_j}^{i_y}$, defined as $\Delta\Theta_{\boldsymbol{h}_i,\boldsymbol{h}_j}^{i_y} = \max\{\boldsymbol{\theta}_{\boldsymbol{h}_i,\boldsymbol{h}_j}^{i_y}\}$, indicates the maximal difference of system behavior in terms of the i_yth system output between $\boldsymbol{H}_{\boldsymbol{h}_i}$ and $\boldsymbol{H}_{\boldsymbol{h}_j}$. That means, if $\Delta\Theta_{\boldsymbol{h}_i,\boldsymbol{h}_j}^{i_y}$ is sufficiently small, the system can be regarded as a linear one in the two subregions and thus the two hypercube regions can be combined together as a hypercuboid region in terms of the i_yth system output, and vice versa.

Considering the system has n_y outputs, the total n_y largest included angles can be computed for the n_y I/O surfaces, which are written in the following vector form:

$$\Delta\mathbf{\Theta}_{\boldsymbol{h}_i,\boldsymbol{h}_j} = [\Delta\Theta_{\boldsymbol{h}_i,\boldsymbol{h}_j}^{1}, \Delta\Theta_{\boldsymbol{h}_i,\boldsymbol{h}_j}^{2}, ..., \Delta\Theta_{\boldsymbol{h}_i,\boldsymbol{h}_j}^{i_y}, ..., \Delta\Theta_{\boldsymbol{h}_i,\boldsymbol{h}_j}^{n_y}]^{\mathrm{T}}_{n_y\times 1} \tag{9}$$

Given a designer-assigned angle threshold $\boldsymbol{\gamma} = [\gamma^1, \gamma^2, \cdots, \gamma^{n_y}]^{\mathrm{T}}$ where γ^{i_y} $(1 \le i_y \le n_y)$ is the threshold for the i_yth steady-state I/O surface, if

$$\Delta\Theta_{\boldsymbol{h}_i,\boldsymbol{h}_j}^{i_y} < \gamma^{i_y} \quad \text{for all } i_y,\ 1 \le i_y \le n_y \tag{10}$$

then the two hypercube regions, $\boldsymbol{H}_{\boldsymbol{h}_i}$ and $\boldsymbol{H}_{\boldsymbol{h}_j}$, can be merged into a large subregion in which the system can be regarded to be linear under the criteria $\boldsymbol{\gamma}$. The inequality 10 is the basic condition of combining two hypercube regions, which is also the basis of the following hypercuboid subregion combination.

3.3.2. Hypercuboid subregion combination. As shown in Figure 3, suppose the current divided region, which is also called the "original region" for the current division, starts at $\boldsymbol{h}_s = [h_s^1, h_s^2, ..., h_s^{n_u}]^{\mathrm{T}}$ and ends at $\boldsymbol{h}_e = [h_e^1, h_e^2, ..., h_e^{n_u}]^{\mathrm{T}}$, where $1 \le h_s^{i_u}, h_e^{i_u} \le m^{n_u}$ $(1 \le i_u \le n_u)$. The current "considered region" is in the u^c direction

(c can be 1,2, ..., n_u), starts at $\underline{\boldsymbol{h}_c} = [h_s^1, ..., h_e^c + 1, ..., h_s^{n_u}]^{\mathrm{T}}$, and ends at $\overline{\boldsymbol{h}_c} = [h_e^1, ..., h_e^c + 1, ..., h_e^{n_u}]^{\mathrm{T}}$. The original region and the considered region are both hypercuboid regions and contain $N_o = \prod_{i_u=1}^{n_u}(h_e^{i_u} - h_s^{i_u} + 1)$ and $N_c = \prod_{i_u=1, i_u\neq c}^{n_u}(h_e^{i_u} - h_s^{i_u} + 1)$ hypercubes, respectively. The following two conditions determine whether the considered region can be combined with the original region:

1. Any two different hypercubes in the considered region satisfy the combination condition (10), namely

$$\Delta\Theta_{\boldsymbol{h}_{ic},\boldsymbol{h}_{jc}}^{i_y} < \gamma^{i_y} \qquad \text{for all } i_y,\ 1 \le i_y \le n_y \tag{11}$$

in which the subscripts ic and jc mean $\Delta\Theta_{\boldsymbol{h}_{ic},\boldsymbol{h}_{jc}}^{i_y}$ are computed by using the hypercubes in the considered region.

2. Any hypercube in the original region and any one in the considered region satisfy the combination condition in eq 10, namely

$$\Delta\Theta_{\boldsymbol{h}_{io},\boldsymbol{h}_{jc}}^{i_y} < \gamma^{i_y} \qquad \text{for all } i_y,\ 1 \le i_y \le n_y \tag{12}$$

in which the subscript io means one hypercube comes from the original region.

Writing $\Theta_{\boldsymbol{h}_{io},\boldsymbol{h}_{jc}}^{i_y}$ and $\Theta_{\boldsymbol{h}_{io},\boldsymbol{h}_{jc}}^{i_y}$ in vector form of eq 9, we can finally construct the following two included angle matrices

$$\boldsymbol{\Lambda}_{cc} = [\Delta\boldsymbol{\Theta}_{\boldsymbol{h}_{ic},\boldsymbol{h}_{jc}}]_{n_y\times(N_c(N_c-1)/2)}, \qquad \boldsymbol{\Lambda}_{oc} = [\Delta\boldsymbol{\Theta}_{\boldsymbol{h}_{io},\boldsymbol{h}_{jc}}]_{n_y\times(N_oN_c)} \tag{13}$$

where $\Delta\boldsymbol{\Theta}_{\boldsymbol{h}_{io},\boldsymbol{h}_{jc}}$ and $\Delta\boldsymbol{\Theta}_{\boldsymbol{h}_{ic},\boldsymbol{h}_{jc}}$ are formed according to the method to compute $\Delta\boldsymbol{\Theta}_{\boldsymbol{h}_i,\boldsymbol{h}_j}$ in eq 9.

Define

$$\boldsymbol{\Lambda} = [\boldsymbol{\Lambda}_{oo}, \boldsymbol{\Lambda}_{oc}]_{n_y\times(N_c(N_c-1)/2+N_oN_c)} \tag{14}$$

Calculating the maximal values of $\boldsymbol{\Lambda}$ in a row, we can get the included angle vector

$$\Delta\bar{\boldsymbol{\Theta}} = [\Delta\bar{\Theta}^1, \Delta\bar{\Theta}^2, ..., \Delta\bar{\Theta}^{n_y}]_{n_y\times 1}^{\mathrm{T}} \tag{15}$$

Thus, similar to the combination condition in eq 10, the criterion to combine two hypercuboid regions is

$$\Delta\bar{\Theta}^{i_y} < \gamma^{i_y} \quad \text{for all } i_y, \quad 1 \le i_y \le n_y \tag{16}$$

This criterion (eq 16) provides the hypercuboid region combination condition for the main procedure of the division method.

3.4. Main procedure: division of operating space.

Without loss of generality, we introduce the division method for a two-input two-output (TITO) two-block cascade system. If there exist more system inputs and outputs, the method can be directly extended to accommodate such situations.

The main procedure of the normal vector included angle division method consists of the following steps.

1. Parameter setting.
   Choose the division number vertor $\boldsymbol{m} = [m^1, m^2]^{\mathrm{T}}$ and the threshold value vector $\boldsymbol{\gamma} = [\gamma^1, \gamma^2]^{\mathrm{T}}$. Partition the input space into $m^1 \times m^2 \times 2$ simplices by subprocedure1.
2. Initialization.
   (a) Set $\boldsymbol{h}_s = [1, 1]^{\mathrm{T}}$, $\boldsymbol{h}_e = \boldsymbol{h}_s$.
   (b) Define the block direction flag vector $\boldsymbol{flag}_{bd} = [flag_{bd}^1, flag_{bd}^2]^{\mathrm{T}}$ and set $\boldsymbol{flag}_{bd} = [0, 0]^{\mathrm{T}}$. The vector $\boldsymbol{flag}_{bd}$ is used to block the division direction. $flag_{bd}^{i_u} = 1$ means the u^{i_u} direction is blocked, and vice versa.
   (c) Define the updating direction flag vector $\boldsymbol{flag}_{ud} = [flag_{ud}^1, flag_{ud}^2]^{\mathrm{T}}$ and set $\boldsymbol{flag}_{ud} = [0, 0]^{\mathrm{T}}$. The vector $\boldsymbol{flag}_{ud}$ indicates the division direction in the previous steps. $\boldsymbol{flag}_{bd}$ and $\boldsymbol{flag}_{ud}$ are used together to determine the following division direction.
3. Determine the division direction.
   (a) If all directions are blocked ($\boldsymbol{flag}_{bd} = [1, 1]^{\mathrm{T}}$), go to Step 5.
   (b) Otherwise, check each direction from u^1 to u^{n_u} in sequence. If $flag_{bd}^{i_u} = 0$ and $flag_{ud}^{i_u} = 0$, then go to Step 4 to conduct division in the direction u^{i_u}.
4. Division in the direction u^{i_u}.
   (a) If $h_e^{i_u} = m^{i_u}$ or some hypercube in the considered region has been classified, set $flag_{bd}^{i_u} = 1$ and $flag_{ud}^{i_u} = 1$. If the vector $\boldsymbol{flag}_{ud} = [1, 1]^{\mathrm{T}}$, set $\boldsymbol{flag}_{ud} = [0, 0]^{\mathrm{T}}$. Go to Step 3.
   (b) Otherwise, calculate the included angle vector $\Delta\bar{\boldsymbol{\Theta}}$ for the original region and the considered region.
      i. If $\Delta\bar{\boldsymbol{\Theta}}$ satisfies the hypercuboid combination condition (16), set $h_e^{i_u} = h_e^{i_u} + 1$. If $h_e^{i_u} = m^{i_u}$, set $flag_{bd}^{i_u} = 1$.
      ii. If $\Delta\bar{\boldsymbol{\Theta}}$ does not satisfy the hypercuboid combination condition (eq 16), set $flag_{bd}^{i_u} = 1$.
      iii. To update the division direction, set $flag_{ud}^{i_u} = 1$. If the vector $\boldsymbol{flag}_{ud} = [1, 1]^{\mathrm{T}}$, set $\boldsymbol{flag}_{ud} = [0, 0]^{\mathrm{T}}$. Go to Step 3.
5. Subregion determination and reinitialization.
   (a) At this step, total $(h_e^1 - h_s^1 + 1) \times (h_e^2 - h_s^2 + 1)$ hypercubes have been classified to one group and the corresponding hypercuboid region $\{\boldsymbol{u} \mid \underline{u}^1 + (h_s^1 - 1)\delta^1 \le u^1 \le \underline{u}^1 + h_e^1\delta^1, \underline{u}^2 + (h_s^2 - 1)\delta^2 \le u^2 \le \underline{u}^2 + h_e^2\delta^2\}$ forms an independent subregion. The center of the subregion is chosen as the operating point, around which a linear submodel can be identified.
   (b) If there are remainder hypercubes to be classified, find the index of unclassified hypercube $\boldsymbol{h} = [h^1,h^2]^{\mathrm{T}}$ with the minimum coordinate h^1 in the u^1 direction (if $h^1 = m^1$, consider the minimum coordinate h^2 in the u^2 direction, and so forth). Update $\boldsymbol{h}_s = \boldsymbol{h} = [h^1,h^2]^{\mathrm{T}}$, $\boldsymbol{h}_e = \boldsymbol{h}_s$, and $\boldsymbol{flag}_{bd} = [0, 0]^{\mathrm{T}}$. Go to Step 3.
   (c) If all hypercubes have been classified, the subregion division of the two-block cascade system is completed. Go to Step 6.
6. Remove redundant submodels.
   Suppose N_m subregions are classified by using the above steps. Possibly, not all corresponding submodels are necessary. This means, although the N_m subregions are classified into different parts, the studied system are likely to have similar static gain in some of these subregions. Namely, some submodels can be exempt. Suppose the operating points of two subregions belong to the two hypercubes $\boldsymbol{H}_{h_i}$ and $\boldsymbol{H}_{h_j}$ in the input space, respectively. If the corresponding included angle vector $\Delta\boldsymbol{\Theta}_{\boldsymbol{h}_i,\boldsymbol{h}_j}$ satisfies the hypercube combination criterion in eq 10, the system static gains in the two subregions are close to one another and only one of the two corresponding submodels is to be contained in the model bank for control. By pairwise comparing the previously obtained subregions, the minimal model bank is finally constructed.

Remark 3.1: This division method is naturally applicable to SISO systems. In such a system, the normal vector included angle has the same effect with the slope included angle. This means that the slope included angle division method proposed by Du et al.[9] is a special case of the discussed method.

Remark 3.2: The above steps start the division procedure from the hypercube region with the minimum values of system input. The users can change the starting region by initializing $\boldsymbol{h}_s$ as other values in step 2(a). Different starting regions may lead to different dividing results. These results are all "optimal" in the sense of the prescribed angle threshold $\boldsymbol{\gamma}$.

Remark 3.3: The computational burden of this division method is significant. It strongly depends on the number of system inputs n_u, because the number of simplices $(\prod_{i=1}^{n_u} m^{i_u} \times n_u!)$ is highly related to n_u. The method discussed is quite computationally efficient when there are 2 or 3 system inputs. This is not a very limiting constraint as numerous technological processes, e.g. chemical reactors[10,11,13] and distillation columns,[12] satisfy that condition. On the other hand, for systems with a large number of inputs, the division parameters m^{i_u} (i_u = 1,..., n_u) can be employed to balance the computational burden and the finer division. Decreasing m^{i_u} will reduce the computational burden at the cost of rougher divising, and vice versa.

Remark 3.4: The above method is efficient for many common industrial processes, but it may become very time-consuming when the I/O surfaces in some outputs are extremely complicated. In such a case, one can first carry out the above procedure for each I/O surface, respectively. In particular, for the I/O surface in which the output changes periodically with the system inputs, one just needs to carry out the division procedure for some one-period operating subregion and then map the dividing results to the remaining operating space accordingly. Next, the resulting subregion boundaries from all the I/O surfaces can be together employed to finally divide the operating space.

4. MULTILINEAR MODEL PREDICTIVE CONTROL

In this part, the multilinear extended nonminimal state space (MENMSS) model is initially formed to reject process disturbance and deal with model mismatch. Then, an improved multilinear model predictive control algorithm is designed with the proposed trajectory scheduling technique.

4.1. Multilinear extended nonminimal state space model.

Consider the multilinear model in the form of eq 3. By adding the back shift operator Δ to eq 3, eq 3 is expressed as

$$\Delta\boldsymbol{y}_k + \boldsymbol{F}_1^i\Delta\boldsymbol{y}_{k-1} + ... + \boldsymbol{F}_n^i\Delta\boldsymbol{y}_{k-n} = \boldsymbol{H}_1^i\Delta\boldsymbol{u}_{k-1} + \boldsymbol{H}_2^i\Delta\boldsymbol{u}_{k-2} + ... + \boldsymbol{H}_n^i\Delta\boldsymbol{u}_{k-n} + \Delta\boldsymbol{\varepsilon}_k, \quad \boldsymbol{s} \in \Phi_i \tag{17}$$

Based on the works in Wang[23] and Zhang,[24] the nonminimum state vector $\Delta\boldsymbol{x}_k$ is chosen as

$$\Delta\boldsymbol{x}_k = [\Delta\boldsymbol{y}_k^{\mathrm{T}}, \Delta\boldsymbol{y}_{k-1}^{\mathrm{T}}, ..., \Delta\boldsymbol{y}_{k-n+1}^{\mathrm{T}}, \Delta\boldsymbol{u}_{k-1}^{\mathrm{T}}, \Delta\boldsymbol{u}_{k-2}^{\mathrm{T}}, ..., \Delta\boldsymbol{u}_{k-n+1}^{\mathrm{T}}]^{\mathrm{T}} \tag{18}$$

where $\Delta\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ and $n_x = n_y \times n + n_u \times (n-1)$.

Then, eq 17 can be transformed into the following state space model:

$$\begin{cases} \Delta\boldsymbol{x}_{k+1} = \boldsymbol{A}_N^i\Delta\boldsymbol{x}_k + \boldsymbol{B}_N^i\Delta\boldsymbol{u}_k + \boldsymbol{B}_{N,\varepsilon}\Delta\boldsymbol{\varepsilon}_{k+1} \\ \Delta\boldsymbol{y}_k = \boldsymbol{C}_N^i\Delta\boldsymbol{x}_k \end{cases}, \quad \boldsymbol{s} \in \Phi_i \tag{19}$$

where

$$\boldsymbol{A}_N^i = \begin{bmatrix} -\boldsymbol{F}_1^i & \cdots & -\boldsymbol{F}_{n-1}^i & -\boldsymbol{F}_n^i & \boldsymbol{H}_2^i & \cdots & \boldsymbol{H}_{n-1}^i & \boldsymbol{H}_n^i \\ \boldsymbol{I}_{n_y\times n_y} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{I}_{n_y\times n_y} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{n_u\times n_u} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{I}_{n_u\times n_u} & \boldsymbol{0} \end{bmatrix}_{n_x\times n_x}$$

$$\boldsymbol{B}_N^i = [(\boldsymbol{H}_1^i)^{\mathrm{T}} \;\; \boldsymbol{0} \;\; \cdots \;\; \boldsymbol{0} \;\; \boldsymbol{I}_{n_u\times n_u} \;\; \boldsymbol{0} \;\; \cdots \;\; \boldsymbol{0}]_{n_x\times n_u}^{\mathrm{T}}$$

$$\boldsymbol{B}_{N,\varepsilon} = [\boldsymbol{I}_{n_y\times n_y} \;\; \boldsymbol{0} \;\; \cdots \;\; \boldsymbol{0} \;\; \boldsymbol{0} \;\; \boldsymbol{0} \;\; \cdots \;\; \boldsymbol{0}]_{n_x\times n_y}^{\mathrm{T}}$$

$$\boldsymbol{C}_N^i = [\boldsymbol{I}_{n_y\times n_y} \;\; \boldsymbol{0} \;\; \cdots \;\; \boldsymbol{0} \;\; \boldsymbol{0} \;\; \boldsymbol{0} \;\; \cdots \;\; \boldsymbol{0}]_{n_y\times n_x}$$

$\boldsymbol{0}$ and $\boldsymbol{I}$ are a zero matrix and a unit matrix with appropriate dimensions, respectively.

Define the output tracking error as

$$\boldsymbol{e}_k = \boldsymbol{y}_k - \boldsymbol{r}_k \tag{20}$$

where $\boldsymbol{r}_k$ is the set-point at the instant k. By combining eqs 19 and 20, we derive $\boldsymbol{e}_{k+1}$ as follows:

$$\boldsymbol{e}_{k+1} = \boldsymbol{e}_k + C_N^iA_N^i\Delta\boldsymbol{x}_k + C_N^iB_N^i\Delta\boldsymbol{u}_k + \Delta\boldsymbol{\varepsilon}_{k+1} - \Delta\boldsymbol{r}_{k+1}, \quad \boldsymbol{s} \in \Phi_i \tag{21}$$

By combining the state vector $\Delta\boldsymbol{x}_k$ and the output tracking error $\boldsymbol{e}_k$, the extended nonminimum state vector is defined as[24,25]

$$\boldsymbol{z}_k = \begin{bmatrix} \Delta\boldsymbol{x}_k \\ \boldsymbol{e}_k \end{bmatrix} \tag{22}$$

where $\boldsymbol{z}_k \in \mathbb{R}^{n_z}$ and $n_z = n_y \times (n+1) + n_u \times (n-1)$.

Based on eqs 19, 21, and 22, we get the MENMSS model:

$$\boldsymbol{z}_{k+1} = \boldsymbol{A}^i\boldsymbol{z}_k + \boldsymbol{B}^i\Delta\boldsymbol{u}_k + C^i\Delta\boldsymbol{r}_{k+1} + \boldsymbol{B}_\varepsilon\Delta\boldsymbol{\varepsilon}_{k+1}, \quad \boldsymbol{s} \in \Phi_i \tag{23}$$

where

$$\boldsymbol{A}^i = \begin{bmatrix} \boldsymbol{A}_N^i & \boldsymbol{0}_{n_x\times n_y} \\ \boldsymbol{C}_N^i\boldsymbol{A}_N^i & \boldsymbol{I}_{n_y\times n_y} \end{bmatrix}; \boldsymbol{B}^i = \begin{bmatrix} \boldsymbol{B}_N^i \\ \boldsymbol{B}_N^i\boldsymbol{A}_N^i \end{bmatrix}; \boldsymbol{C}^i = \begin{bmatrix} \boldsymbol{0}_{n_x\times n_y} \\ -\boldsymbol{I}_{n_y\times n_y} \end{bmatrix}; \boldsymbol{B}_\varepsilon = \begin{bmatrix} \boldsymbol{B}_{N,\ \varepsilon} \\ \boldsymbol{I}_{n_y\times n_y} \end{bmatrix} \tag{24}$$

Note, the frequently used "DMC type" disturbance model[11,16] is adopted in the following control algorithm. Over the prediction horizon in the following control algorithm, the error term $\boldsymbol{\varepsilon}_k$ and its increment $\Delta\boldsymbol{\varepsilon}_k$ are assumed to be a constant and zero, respectively. Therefore, the effect of the error term, the term $\boldsymbol{B}_\varepsilon \Delta \boldsymbol{\varepsilon}_{k+1}$ in eq 23, is exempt from the following algorithm.

4.2. Predictive control based on the MENMSS model.

4.2.1. MPC optimization problem. At each sampling instant, MPC determines the future sequence of input increments

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

by solving the following problem with constraints

$$\begin{aligned} \min_{\substack{\Delta \boldsymbol{u}_{k|k}, \ldots, \\ \Delta \boldsymbol{u}_{k+M-1|k}}} \quad & \sum_{j=1}^{P} \|\boldsymbol{z}_{k+j|k}\|^2_{\boldsymbol{w}_{z,k+j|k}} + \sum_{j=0}^{M-1} \|\Delta \boldsymbol{u}_{k+j|k}\|^2_{\boldsymbol{w}_{\Delta u,k+j|k}} \\ \text{s. t.} \quad & \boldsymbol{z}_{k+j|k} = \boldsymbol{A}^i \boldsymbol{z}_{k+j-1|k} + \boldsymbol{B}^i \Delta \boldsymbol{u}_{k+j-1|k} \\ & + C^i \Delta \boldsymbol{r}_{k+j|k}, \text{ if } \boldsymbol{s}_{k+j|k} \in \Phi_i (i = 1, \ldots, N_m) \\ & \Delta \boldsymbol{u}_{k+j|k} = 0, \quad \text{if } j \geq M \\ & \boldsymbol{u}_{\min} \leq \boldsymbol{u}_{k+j|k} \leq \boldsymbol{u}_{\max} \\ & \Delta \boldsymbol{u}_{\min} \leq \Delta \boldsymbol{u}_{k+j|k} \leq \Delta \boldsymbol{u}_{\max} \end{aligned} \tag{25}$$

where the norm is defined as $\|\boldsymbol{x}\|^2_W \triangleq \boldsymbol{x}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{x}$; P is the prediction horizon; M is the control horizon; $\boldsymbol{w}_{z,k+j|k} \geq \boldsymbol{0}_{n_z \times n_z}$, $\boldsymbol{w}_{\Delta u,k+j|k} \geq \boldsymbol{0}_{n_u \times n_u}$ are the weights of the minimized cost-function; $\boldsymbol{u}_{\min} \in \mathbb{R}^{n_u}$, $\boldsymbol{u}_{\max} \in \mathbb{R}^{n_u}$, $\Delta \boldsymbol{u}_{\min} \in \mathbb{R}^{n_u}$, and $\Delta \boldsymbol{u}_{\max} \in \mathbb{R}^{n_u}$ are the constraints imposed on the magnitude and on the increments of system inputs, respectively. Here, we ignore output constraints for simplicity. Interested readers can refer to Wang,[23] which shows output constraints can be easily integrated into the problem (eq 25). At each sampling instant, M future control increments are calculated from the MPC optimization problem (eq 25); 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.

Because different local linear models are used to characterize the process in different subregions Φ_i, the predicted states $\boldsymbol{z}_{k+j|k}$ are essentially nonlinear functions of the optimized online control increments $\Delta \boldsymbol{U}_k$. Directly solving the complex nonlinear optimization problem (eq 25) for $\Delta \boldsymbol{U}_k$ is computationally difficult and will lead to a heavy computational burden. The more practical approach is first to design local MPC controllers from the local linear models and then to combine these local controllers into a global one.[2,17] Based on this idea, a computational efficient multilinear MPC algorithm with trajectory scheduling (MMPC-TS) is developed in the following subsection.

4.2.2. Multilinear MPC algorithm with trajectory scheduling. In the proposed MMPC-TS algorithm, the multilinear model is scheduled as different linear models over the predictive horizon by using a predicted scheduling variable trajectory. Then, the predicted states $\boldsymbol{z}_{k+j|k}$ become the linear functions of the future control increments $\Delta \boldsymbol{U}_k$, and the nonlinear programming problem (eq 25) is simplified into a quadratic programming (QP) problem. So, $\Delta \boldsymbol{U}_k$ can be easily computed (the computational burden of quadratic optimization is much lower than that of general nonlinear optimization and the global optimal solution is always found). For accurate scheduling, the aforementioned trajectory scheduling and the resulting QP optimization may be repeated several times in internal iterations at each sampling instant.

Suppose a set of local linear controllers are designed based on the minimum model bank. To simplify design of the following global controller, we choose the same predictive horizon P and control horizon M for these local controllers, whereas the weights are separately tuned, donated as $\boldsymbol{w}^i_{z,k+j|k}$ and $\boldsymbol{w}^i_{\Delta u,k+j|k}$ $(i = 1, \ldots, N_m)$, respectively. Then, the proposed MMPC-TS algorithm is developed as follows.

Define the predicted scheduling variable trajectory at sampling instant k as

$$\boldsymbol{S}_k = [\boldsymbol{s}_{k+1|k}^{\mathrm{T}} \boldsymbol{s}_{k+2|k}^{\mathrm{T}} \ldots \boldsymbol{s}_{k+P|k}^{\mathrm{T}}]^{\mathrm{T}} \tag{26}$$

and define

$$\boldsymbol{Z}_k = \begin{bmatrix} \boldsymbol{z}_{k+1|k} \\ \boldsymbol{z}_{k+2|k} \\ \vdots \\ \boldsymbol{z}_{k+P|k} \end{bmatrix}, \quad \Delta \boldsymbol{R}_k = \begin{bmatrix} \Delta \boldsymbol{r}_{k+1|k} \\ \Delta \boldsymbol{r}_{k+2|k} \\ \vdots \\ \Delta \boldsymbol{r}_{k+P|k} \end{bmatrix} \tag{27}$$

Taking into account the MENMSS model (eq 23), the predicted state trajectory can be derived as

$$\boldsymbol{Z}_k = \boldsymbol{F}_k \boldsymbol{z}_k + \boldsymbol{G}_k \Delta \boldsymbol{U}_k + \boldsymbol{\Psi}_k \Delta \boldsymbol{R}_k \tag{28}$$

where $\boldsymbol{F}_k$, $\boldsymbol{G}_k$, and $\boldsymbol{\Psi}_k$ are parameter matrices

$$\boldsymbol{F}_k = \begin{bmatrix} \boldsymbol{A}_{k+1|k} \\ \boldsymbol{A}_{k+2|k} \boldsymbol{A}_{k+1|k} \\ \vdots \\ \prod_{j=P}^{1} \boldsymbol{A}_{k+j|k} \end{bmatrix}_{n_z P \times n_z} \tag{29}$$

$$\boldsymbol{G}_k = \begin{bmatrix} \boldsymbol{B}_{k+1|k} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{A}_{k+2|k} \boldsymbol{B}_{k+1|k} & \boldsymbol{B}_{k+2|k} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \prod_{j=M+1}^{2} \boldsymbol{A}_{k+j|k} \boldsymbol{B}_{k+1|k} & \prod_{j=M+1}^{3} \boldsymbol{A}_{k+j|k} \boldsymbol{B}_{k+2|k} & \cdots & \boldsymbol{A}_{k+M+1|k} \boldsymbol{B}_{k+M|k} \\ \vdots & \vdots & \ddots & \vdots \\ \prod_{j=P}^{2} \boldsymbol{A}_{k+j|k} \boldsymbol{B}_{k+1|k} & \prod_{j=P}^{3} \boldsymbol{A}_{k+j|k} \boldsymbol{B}_{k+2|k} & \cdots & \prod_{j=P}^{M+1} \boldsymbol{A}_{k+j|k} \boldsymbol{B}_{k+M|k} \end{bmatrix}_{n_z P \times n_u M} \tag{30}$$

$$\boldsymbol{\Psi}_k = \begin{bmatrix} \boldsymbol{C}_{k+1|k} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{A}_{k+2|k} \boldsymbol{C}_{k+1|k} & \boldsymbol{C}_{k+2|k} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \prod_{j=P}^{2} \boldsymbol{A}_{k+j|k} \boldsymbol{C}_{k+1|k} & \prod_{j=P}^{3} \boldsymbol{A}_{k+j|k} \boldsymbol{C}_{k+2|k} & \cdots & \boldsymbol{C}_{k+P|k} \end{bmatrix}_{n_z P \times n_y P} \tag{31}$$

Based on the scheduling variable $\boldsymbol{s}_{k+j|k} (j = 1, \ldots, P)$, the model matrices $\boldsymbol{A}_{k+j|k}$, $\boldsymbol{B}_{k+j|k}$, and $\boldsymbol{C}_{k+j|k}$ are chosen from the obtained MENMSS model. For example, if $\boldsymbol{s}_{k+j|k} \in \Phi_i$ $(i = 1, \ldots, N_m)$, then $\boldsymbol{A}_{k+j|k} = \boldsymbol{A}^i$, $\boldsymbol{B}_{k+j|k} = \boldsymbol{B}^i$, and $\boldsymbol{C}_{k+j|k} = \boldsymbol{C}^i$.

Taking into account the derived state trajectory (eq 28) and considering that $\boldsymbol{U}_k = \boldsymbol{L}\Delta\boldsymbol{U}_k + \boldsymbol{U}_{k-1}$ where

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{I}_{n_u\times n_u} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{I}_{n_u\times n_u} & \boldsymbol{I}_{n_u\times n_u} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{I}_{n_u\times n_u} & \boldsymbol{I}_{n_u\times n_u} & \cdots & \boldsymbol{I}_{n_u\times n_u} \end{bmatrix}_{n_uM\times n_uM},$$

$$\boldsymbol{U}_{k-1} = \begin{bmatrix} \boldsymbol{u}_{k-1} \\ \boldsymbol{u}_{k-1} \\ \vdots \\ \boldsymbol{u}_{k-1} \end{bmatrix}_{n_uM\times 1}$$

the nonlinear optimization problem (eq 25) is simplified as the following QP problem

$$\min_{\Delta\boldsymbol{U}_k} \|\boldsymbol{F}_k\boldsymbol{z}_k + \boldsymbol{G}_k\Delta\boldsymbol{U}_k + \boldsymbol{\Psi}_k\Delta\boldsymbol{R}_k\|^2_{\boldsymbol{W}_{z,k}} + \|\Delta\boldsymbol{U}_k\|^2_{\boldsymbol{W}_{\Delta u,k}}$$
$$s.\ t.\ \boldsymbol{U}_l \le \boldsymbol{U}_{k-1} + \boldsymbol{L}\Delta\boldsymbol{U}_k \le \boldsymbol{U}_u$$
$$\Delta\boldsymbol{U}_l \le \Delta\boldsymbol{U}_k \le \Delta\boldsymbol{U}_u \tag{32}$$

where the vectors of length n_{uM} are

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

the matrix $\boldsymbol{W}_{z,k} = \text{blockdiag}\,\{\boldsymbol{w}_{z,k+1|k}, \boldsymbol{w}_{z,k+2|k}, ..., \boldsymbol{w}_{z,k+P|k}\}$ and the matrix $\boldsymbol{W}_{\Delta u,k} = \text{blockdiag}\{\boldsymbol{w}_{\Delta u,k|k}, \boldsymbol{w}_{\Delta u,k+1|k}, ..., \boldsymbol{w}_{\Delta u,k+M-1|k}\}$. Similar with the model matrices $(\boldsymbol{A}_{k+j|k}, \boldsymbol{B}_{k+j|k}, \boldsymbol{C}_{k+j|k})$, $\boldsymbol{w}_{z,k+j|k}$ $(1 \le j \le P)$ and $\boldsymbol{w}_{\Delta u,k+j|k}$ $(0 \le j \le M-1)$ are selected from the designed local controllers based on the scheduling variable $s_{k+j|k}$ (if $s_{k+j|k} \in \Phi_i$, $\boldsymbol{w}_{z,k+j|k} = \boldsymbol{w}^i_{z,k+j|k}$, and $\boldsymbol{w}_{\Delta u,k+j|k} = \boldsymbol{w}^i_{\Delta u,k+j|k}$).

If the scheduling variable trajectory $\boldsymbol{S}_k$ is known at sampling instant k, the nonlinear MPC optimization problem (eq 25) can be accurately transformed into the QP problem (eq 32). Unfortunately, the trajectories $\boldsymbol{S}_k$ are generally unknown because they are usually functions of optimized variables. Inaccurate trajectory $\boldsymbol{S}_k$ will deteriorate control performance, which has a similar effect with model mismatch. To predict $\boldsymbol{S}_k$ as accurately as possible, the trajectory scheduling (eq 28) and the resulting QP optimization are repeated several times in internal iterations. For convenience, index the related vectors and matrices at the tth iteration with superscript t, such as the vector $\boldsymbol{S}^t_k$, the matrix $\boldsymbol{F}^t_k$, and so on.

The proposed MMPC-TS algorithm is summarized as the following steps:

1. Measure the system output $\boldsymbol{y}_k$. Form the ENMSS state space vector $\boldsymbol{z}_k$ according to eqs 18, 20, and 22.
2. Set the index t to 1. Initialize the scheduling variable trajectory $\boldsymbol{S}^1_k = [(\boldsymbol{s}^1_{k+1|k})^{\mathrm{T}}(\boldsymbol{s}^1_{k+2|k})^{\mathrm{T}} ... (\boldsymbol{s}^1_{bk+P|k})^{\mathrm{T}}]^{\mathrm{T}}$ $(\boldsymbol{s}^1_{k+j|k}, j = 1, ..., P$, are uniformly chosen as $\boldsymbol{u}_{k-1}$ because system inputs are used as the scheduling variables in this study).
3. Calculate the predicted state trajectory $\boldsymbol{Z}^t_k$, i.e., the matrices $\boldsymbol{F}^t_k$, $\boldsymbol{G}^t_k$ and $\boldsymbol{\Psi}^t_k$ are found, as well as the weighting matrices $\boldsymbol{W}^t_{z,k}$ and $\boldsymbol{W}^t_{\Delta u,k}$.
4. The MPC QP problem (eq 32) is solved to find $\Delta\boldsymbol{U}^t_k$ and the vector $\boldsymbol{S}^t_k$ is updated by using $\Delta\boldsymbol{U}^t_k$.
5. Decide whether internal iterations should be terminated. If the trajectory $\Delta\boldsymbol{U}^t_k$ is close to the trajectory at the previous internal iteration, i.e.

$$\|\Delta\boldsymbol{U}^t_k - \Delta\boldsymbol{U}^{t-1}_k\|^2 < \sigma_u \tag{33}$$

or $t > t_{\max}$, then go to Step 6 (σ_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 $\Delta\boldsymbol{U}^t_k$ (i.e., the vector $\Delta\boldsymbol{u}^t_{k|k}$) to the process.

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

Remark 4.1: If $t_{\max}$ is chosen as 1 and $\boldsymbol{s}^1_{k+j|k}$ $(1 \le j \le N_m)$ are set to the same value, the proposed MMPC-TS reduces to the conventional MPC algorithm with hard switching. This means MPC with hard switching is a special case of this developed MMPC-TS algorithm.

Remark 4.2: Compared with the MPC algorithms with soft switching, the proposed MMPC-TS algorithm does not require any tuning parameters for switching, or the complete process dynamic information which is difficult to be obtained in nonlinear systems. Due to the trajectory scheduling technique, the MMPC-TS algorithm still can alleviate output oscillations in comparison with hard switching.

5. CASE STUDY: A MIMO CONTINUOUS STIRRED TANK REACTOR PROCESS

5.1. Process description. Consider a MIMO continuous stirred tank reactor (CSTR) process, which has been modeled by a Hammerstein structure or a Wiener structure in previous studies.[10,11] It consists of an irreversible, exothermic reaction, $A \rightarrow B$, and takes place in a constant volume reactor cooled by a single coolant stream. It can be characterized by the following continuous-time nonlinear equations[10,11]

$$\begin{cases} \dot{C}_A = \dfrac{q}{V}(C_{A0} - C_A) - k_0C_Ae^{-E/RT} \\ \dot{T} = \dfrac{q}{V}(T_0 - T) - \dfrac{\Delta Hk_0}{\rho C_p}C_Ae^{-E/RT} \\ \qquad + \dfrac{\rho_cC_{pc}}{\rho C_pV}q_c(1 - e^{-h_A/\rho_cC_{pc}q_c})(T_{c0} - T) \end{cases} \tag{34}$$

The nominal model parameters are listed in Table 1. The sampling period is T_s = 0.1 min. The process output variables are the concentration of A (C_A) and the temperature (T). The process input variables are the process flow rate (q) and the coolant flow rate (q_c). The control objective is to regulate C_A and T for set-point tracking by manipulating q and q_c.

The variation ranges of the inputs are $q \in [95, 150]$ and $q_c \in [60, 110]$. The variation ranges of the outputs are $C_A \in [0.02, 0.15]$ and $T \in [430, 490]$. Figure 4 and Figure 5 show the steady-state surfaces of the concentration C_A and the temperature T, respectively. This CSTR shows apparent nonlinearity, and thus a single linear controller is not sufficient for the wide operating range.[10,11]

5.2. Division of the operating space. The proposed normal vector included angle method is employed to decompose

Table 1. Nominal Model Parameters of CSTR

Measured product concentration	C_A	0.1 mol L^{-1}
Reactor temperature	T	438.5 K
Coolant flow rate	q_c	103.41 L min^{-1}
Process flow rate	q	100 L min^{-1}
Feed concentration	C_{A0}	1 mol L^{-1}
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^{-1} K^{-1}
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^{-1}
Liquid densities	ρ, ρ_c	1×10^3 g L^{-1}
Specific Heats	C_p, C_{pc}	1 cal g^{-1} K^{-1}

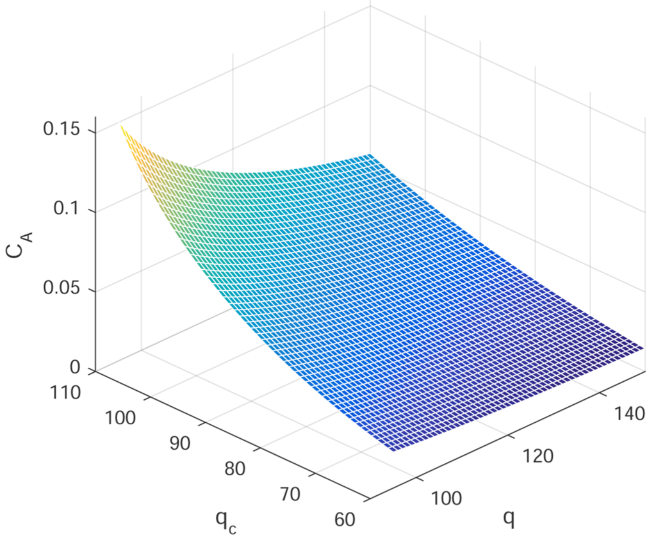


Figure 4. Steady-state surface of the concentration C_A.

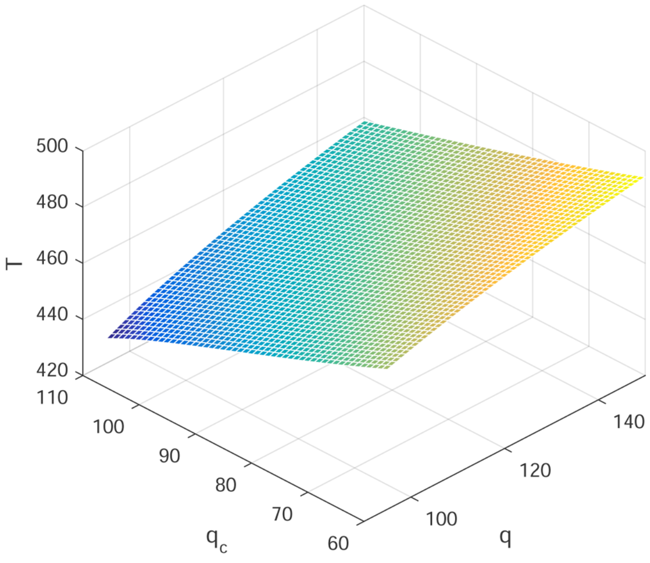


Figure 5. Steady-state surface of the concentration T.

the CSTR system. The operating points (OP) for the CSTR are expressed in the form of (q, q_c, C_A, T). Four cases of division are studied to show the effect of the designer-assigned parameters. The dividing results are depicted in Figure 6.

Case 1. $\boldsymbol{m} = [10, 10]^T$, $\boldsymbol{\gamma} = [0.003, \pi/10]^T$. By using the first five steps of the proposed division method, the operating space is initially decomposed into three subregions, donated as Φ_1, Φ_2, and Φ_3. The corresponding three operating points are OP$_1$ (122.5, 77.5, 0.0303, 468.79), OP$_2$ (106, 102.5, 0.0855, 443.04) and OP$_3$ (133.5, 102.5, 0.0562, 456.83). Next, according to Step 6 of the division method, the operating points OP$_1$ and OP$_3$ are close in the normal vector included angle sense, where the CSTR has similar static gains. Thus, subregions Φ_1 and Φ_3 can share a common submodel (which is selected as the submodel at OP$_1$ in this study). The final model bank should only include the two linear submodels at OP$_1$ and OP$_2$.

Case 2. $\boldsymbol{m} = [30, 30]^T$, $\boldsymbol{\gamma} = [0.003, \pi/10]^T$. In this case, the operating space is also divided into three subregions and the final model bank should also include two submodels at the two operating points OP$_1$(122.5, 76.67, 0.0295, 469.39) and OP$_2$ (104.17, 101.67, 0.0860, 442.55).

Case 3. $\boldsymbol{m} = [60, 60]^T$, $\boldsymbol{\gamma} = [0.003, \pi/10]^T$. As shown in Figure 6, this case has the same dividing results with Case 2.

In the above three cases, the designer-assigned parameters $\boldsymbol{h} = [m^1, m^2]^T$ are selected to be different values, whereas the threshold value vector $\boldsymbol{\gamma}$ is chosen as the same $[0.003, \pi/10]^T$. The three cases have almost the same dividing results: the same number of subregions, nearly the same subregion ranges, and nearly the same operating points. In particular, Case 2 has exactly the same dividing results as Case 3. The larger the numbers of divisions m^1 and m^2 are, the smaller simplices the input region is partitioned into and the more system characteristics the dividing procedure considers. Compared with Case 1, Case 2 and Case 3 obtain the narrower subregion Φ_1 in the q_c direction and the narrower subregion Φ_2 in the q direction. This is because larger m^1 and m^2 lead to finer and more accurate dividing in Case 2 and Case 3, while smaller m^1 and m^2 make the division in Case 1 rougher. Therefore, sufficiently large numbers of divisions m^{i_u} ($i_u = 1, 2, ..., n_u$) are generally suggested for accurate dividing (e.g., $\boldsymbol{m} = [30, 30]^T$ for the CSTR) if the off-line computation time is allowable.

Case 4. $\boldsymbol{m} = [30, 30]^T$, $\boldsymbol{\gamma} = [0.0028, \pi/30]^T$. In this case, the operating space is divided into five subregions as depicted in Figure 6. The subregions Φ_2 and Φ_3 share the same submodel with the subregion Φ_1. The minimal model bank resulted from the proposed division method includes the three submodels at the three operating points: OP$_1$ (113.33, 68.33, 0.0247, 471.69), OP$_3$ (104.17, 101.67, 0.0860, 442.55), and OP$_5$ (140.83, 85, 0.0318, 470.77), respectively.

Case 4 and Case 2 choose the same numbers of divisions ($\boldsymbol{m} = [30, 30]^T$), whereas Case 4 uses a smaller threshold values γ^{i_y} ($i_y = 1, 2, ..., n_y$) than Case 2. The smaller the threshold value γ^{i_y} is, the smaller nonlinearity the dividing procedure allows in each subregion for the i_yth I/O surface, and vice versa. For this reason, the operating space is decomposed into more subregions in Case 4 than in Case 2. In practice, proper threshold values should be chosen according to the prior knowledge of the system and the control performance.

In this study, the dividing result of Case 2 is employed to construct the linear model bank. Two second-order TITO linear submodels in the form of eq 3 are respectively identified at OP$_1$ and OP$_2$, whose parameters are given as follows:

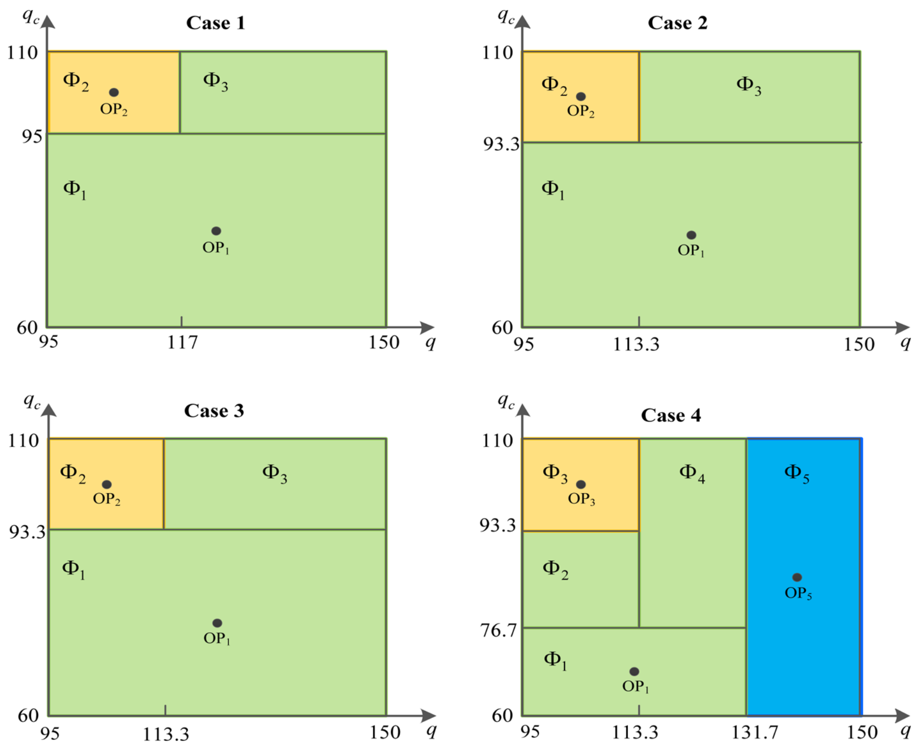


Figure 6. Dividing results of the four cases for the CSTR.

$$F_1^1 = \begin{bmatrix} -0.84371 & 0 \\ 0 & -0.84371 \end{bmatrix},$$
$$F_2^1 = \begin{bmatrix} 3.8045 \times 10^{-2} & 0 \\ 0 & 3.8045 \times 10^{-2} \end{bmatrix},$$
$$H_1^1 = \begin{bmatrix} 2.148 \times 10^{-4} & 1.3291 \times 10^{-4} \\ 2.7358 \times 10^{-2} & -1.3392 \times 10^{-1} \end{bmatrix},$$
$$H_2^1 = \begin{bmatrix} -2.7355 \times 10^{-4} & 4.7247 \times 10^{-5} \\ 5.277 \times 10^{-2} & -4.6077 \times 10^{-3} \end{bmatrix},$$
$$F_1^2 = \begin{bmatrix} -1.5235 & 0 \\ 0 & -1.5235 \end{bmatrix},$$
$$F_2^2 = \begin{bmatrix} 6.4138 \times 10^{-1} & 0 \\ 0 & 6.4138 \times 10^{-1} \end{bmatrix},$$
$$H_1^2 = \begin{bmatrix} 6.1264 \times 10^{-4} & 1.9117 \times 10^{-4} \\ -3.4271 \times 10^{-2} & -1.1993 \times 10^{-1} \end{bmatrix},$$
$$H_2^2 = \begin{bmatrix} -8.1253 \times 10^{-4} & 1.6477 \times 10^{-4} \\ 1.0622 \times 10^{-1} & 3.1286 \times 10^{-2} \end{bmatrix}$$

Figure 7 compares the modeling performance of the two linear submodels and the multilinear model. The linear submodel 1 cannot track the sampled data with high concentration (C_A) and low temperature (T), e.g., the trajectories at around the 1000th sampling point. The submodel 2 even gives negative concentration, which is physically not possible. By comparison, the global multilinear model is very precise, whose trajectories are very close to the sampled data. To sum up, the multilinear model can closely approximate the nonlinear MIMO CSTR process in the entire operating space, with much higher accuracy than a single linear submodel.

5.3. Predictive control of the CSTR.

In this part, based on the obtained linear model bank, the proposed MMPC-TS algorithm is applied to the CSTR for set-point tracking. For comparison, two simple Linear MPC (LMPC) controllers (LMPC1 and LMPC2) are designed based on the linear submodel 1 and submodel 2, respectively. They are employed to demonstrate the necessity of utilizing the multilinear model technique and the effectiveness of the proposed division method. Moreover, a multilinear model predictive controller with hard switching (MMPC-HS) is also designed based on the same multilinear model. It is also compared with MMPC-TS to illustrate the advantages of the trajectory scheduling technique introduced in section 4.2.2.

Initially, the two LMPC controllers are tuned with the parameters $P = 20$, $M = 5$, $w^1_{z,k+j|k} = diag\{10^6, 1, 10^6, 1, 0, 0, 10^6, 1\}$, $w^1_{\Delta u,\,k+j|k} = diag\{1, 1\}$, $w^2_{z,k+j|k} = diag\{10^5, 1, 10^5, 1, 0, 0, 10^5, 1\}$, $w^2_{\Delta u,\,k+j|k} = diag\{1, 1\}$. Constraints on the manipulated variables are $95 \le q \le 150$, $60 \le q_c \le 110$, $-10 \le \Delta q \le 10$, and $-10 \le \Delta q_c \le 10$. Next, these parameters are directly used as the control parameters of the MMPC-HS controller and the proposed MMPC-TS controller. Additional parameters of the MPC-TS controller are $\sigma_u = 1$, $t_{max} = 4$.

Figure 8 depicts the closed-loop performance of the MMPC-TS controller. The process outputs C_A and T follow the set-points quickly and precisely, and the process inputs vary exactly in their predetermined ranges, $q \in [95, 150]$ and $q_c \in [60, 110]$. More importantly, the closed-loop tracking is very smooth, during which neither output chattering nor input chattering happen. The MMPC-TS controller based on the two linear submodels performs quite satisfactorily in the wide operating range. This illustrates two linear submodels and the corresponding divided subregions provide sufficient information for this set-point tracking control, and the combination of the parameters $\boldsymbol{m} = [30, 30]^T$ and $\boldsymbol{\gamma} = [0.003, \pi/10]^T$ is a good choice for the division of the MIMO CSTR.

The comparisons between the MMPC-TS controller and the other two linear MPC controllers (LMPC1, LMPC2) are

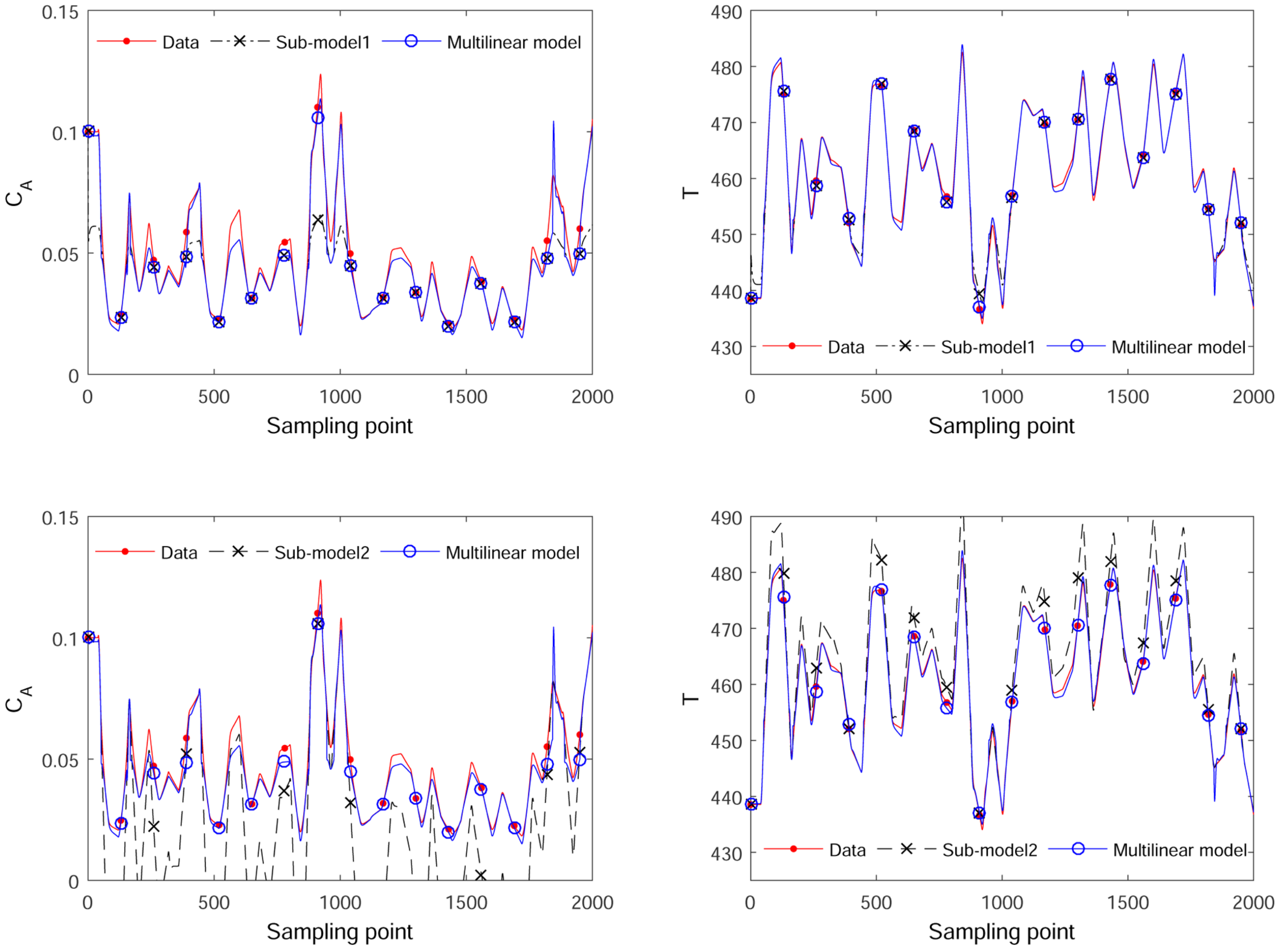


Figure 7. Open-loop model validation of the MIMO CSTR: the linear submodels vs the multilinear model.

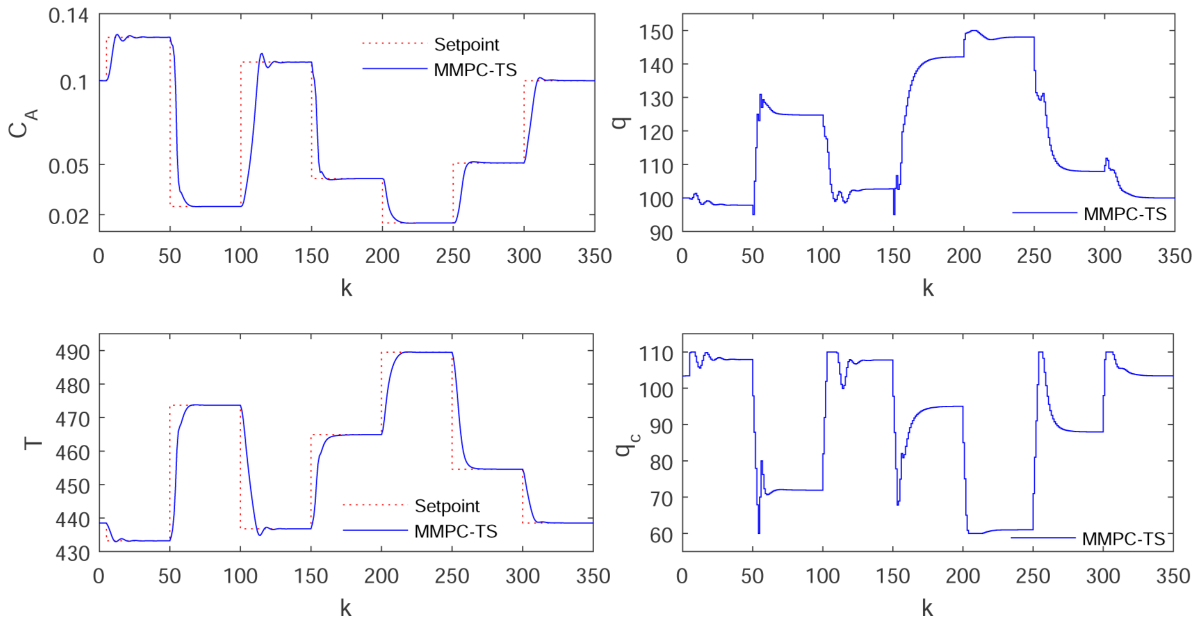


Figure 8. Simulation results of the MIMO CSTR using the MMPC-TS controller.

displayed in Figures 9−12. In Figure 9, the LMPC1 controller performs as well as the MMPC-TS controller at the set-points in the subregions Φ_1 and Φ_3. This illustrates a single submodel is sufficient to characterize the system dynamic in the two

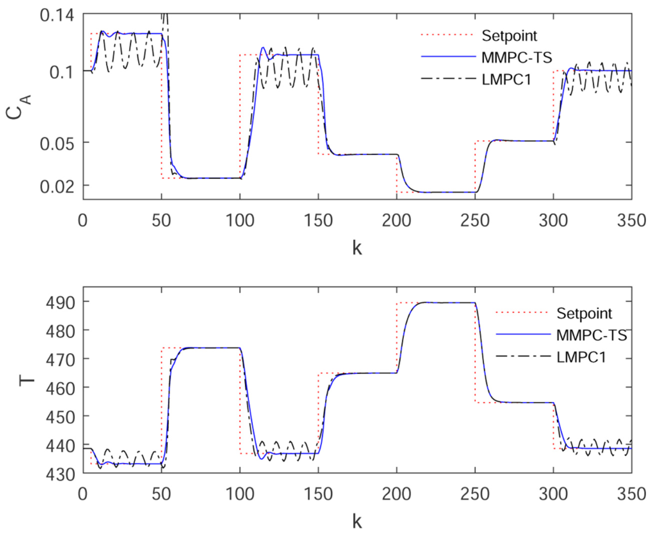


Figure 9. Closed-loop responses: the MMPC-TS controller vs the LMPC1 controller.

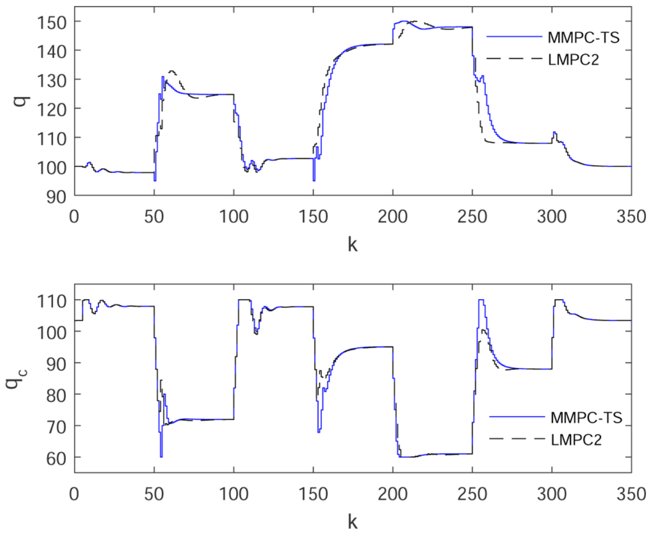


Figure 10. Control inputs: the MMPC-TS controller vs the LMPC1 controller.

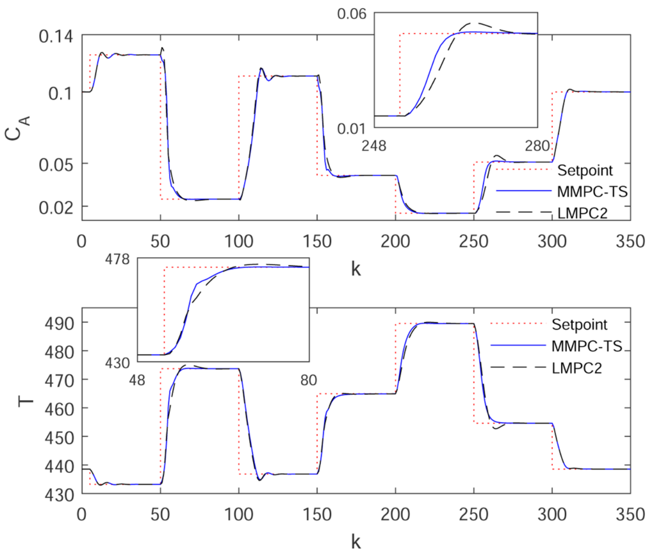


Figure 11. Closed-loop responses: the MMPC-TS controller vs the LMPC2 controller.

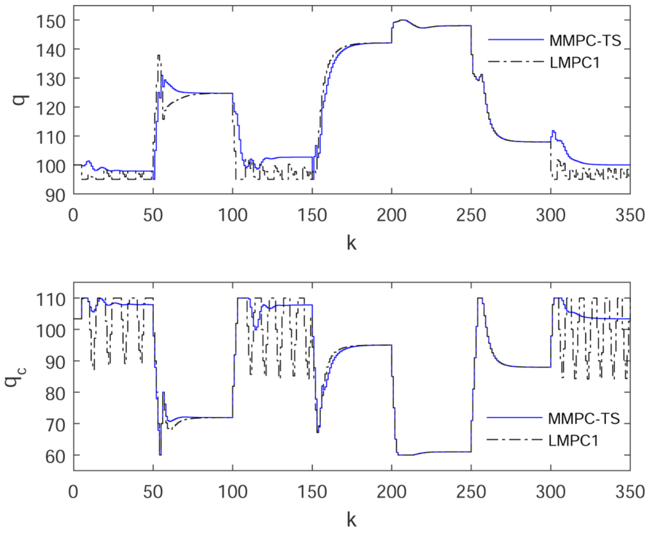


Figure 12. Control inputs: the MMPC-TS controller vs LMPC2 controller.

subregions. However, the LMPC1 controller leads to serious output oscillations at the set-points in the subregion Φ_2. Similarly, in Figure 11, the LMPC2 controller has the same satisfactory performance as the MMPC-TS controller at the set-points in the subregion Φ_2, whereas large overshoots and slow tracking responses happen in other subregions (Φ_1 and Φ_3). As a whole, the linear MPC controllers based on the two linear submodels just perform well in their own subregions. Therefore, two linear submodels are necessary for this nonlinear MIMO CSTR process. This also confirms the effectiveness of the proposed normal vector included angle division method.

Figures 13−14 compare the MMPC-TS controller with the MMPC-HS controller. The MMPC-HS controller can regulate the process outputs to follow their set-points, and it performs as well as the MMPC-TS controller when no submodel switching is activated, such as the responses from $k = 0$ to $k = 50$ and from $k = 201$ to $k = 300$. Nevertheless, during switching submodels, such as the responses from $k = 51$ to $k = 200$ and from $k = 301$ to $k = 350$, the MMPC-HS controller causes unexpected large output oscillations. This is due to the rough and inaccurate switching caused by the hard switching technique in the MMPC-HS controller. By comparison, the proposed trajectory scheduling technique can achieve smooth switching, and its internal iteration mechanism makes the predicted trajectory and the resulting switching in the MMPC-TS controller more accurate than those in the MMPC-HS controller. The total computational time of the MMPC-TS controller is 1.28× that of the MMPC-HS controller. Therefore, the trajectory scheduling technique enables the MMPC-TS controller to give better performance at a slightly higher computational time cost than the MMPC-HS controller. Moreover, it does not require extra complicated weighting parameters/functions used in soft switching. Thanks to the proposed trajectory scheduling technique, the resulting MMPC-TS controller achieves satisfactory balance between smooth control and difficulty in computing control laws and designing controllers.

To compare these controllers more systematically, two kinds of criteria are calculated for the above simulations: the settling time T_s[26] and the overshoot M_p.[27] The settling time T_s refers to

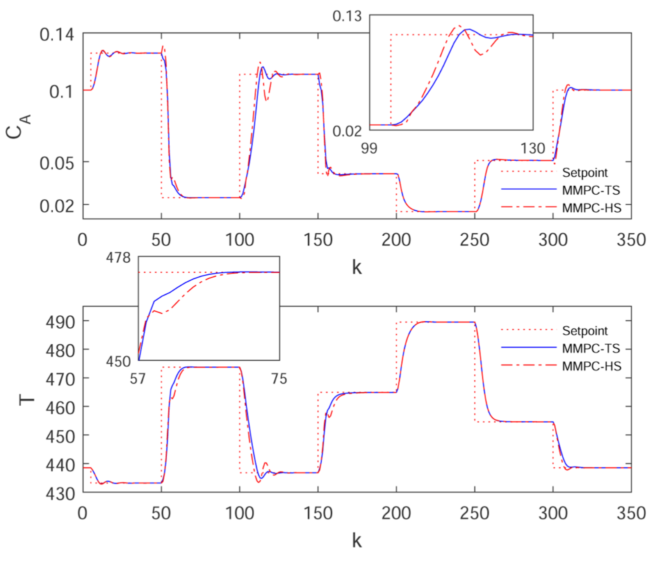


Figure 13. Closed-loop responses: the MMPC-TS controller vs the MMPC-HS controller.

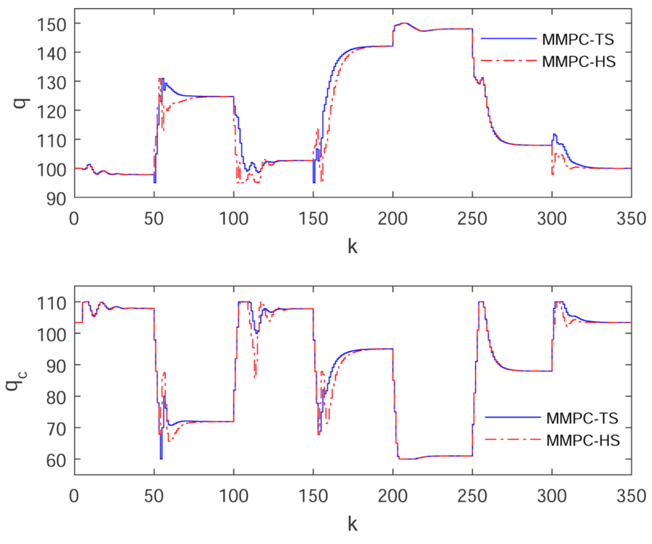


Figure 14. Control inputs: the MMPC-TS controller vs the MMPC-HS controller.

the time from changing set-points to the instant all the controlled variables first reach and thereafter remain within a prescribed percentage $\pm\epsilon$ of their corresponding set-points, which reflects the response speed of the closed-loop system. The overshoot M_p is the percentage between the maximum amount the system overshoots its final value and the final value of the system output, which is an important stability indicator for control. As the simulation consists of seven stages (the set-point changes seven times), the average settling time ($\overline{T}_s$), the average overshoot ($\overline{M}_p$), the maximal settling time (T_s^{max}), and the maximal overshoot (M_p^{max}) of these stages are calculated.

Table 2 gives the values of $\overline{T}_s$ and T_s^{max} (in terms of the three most common prescribed percentages[27]) and the values of $\overline{M}_p$ and M_p^{max} resulted from these MPC controllers. The proposed MMPC-TS has the minimal values in terms of all these criteria among these controllers. It contributes to faster closed-loop responses and higher control precision than other controllers. Therefore, the effectiveness of the proposed division method and the designed MMPC-TS control algorithm is confirmed.

6. CONCLUSION

This study provides a complete multilinear control scheme for MIMO two-block cascade systems. First, a normal vector included angle division method is proposed to decompose the system operating space and determine the minimum linear model bank. It works for the systems with multiple inputs and is an extension of the method proposed by De et al.[9] This method just requires steady-state I/O mappings and thus is more convenient and lower-cost than dynamic information based division methods. Moreover, a novel multilinear MPC algorithm (MMPC-TS) is developed with the trajectory scheduling technique for local controller combination. The trajectory scheduling technique enables MMPC-TS to reduce output oscillations at a slightly higher computational cost compared with the control algorithms with hard switching, and meanwhile it avoids the complex weighting parameters/functions used in soft switching.

The presented division method has a great potential when applied to the classical technological processes with 2 or 3 inputs. For systems with a large number of inputs, users can use the parameters m^{i_u} to balance the computational load and the division precision. Further research should focus on reducing computational load without losing precision.

■ AUTHOR INFORMATION

Corresponding Author

*E-mail: jzhang398-c@my.cityu.edu.hk.

ORCID

Jian Zhang: 0000-0002-1530-6323

Notes

The authors declare no competing financial interest.

■ REFERENCES

(1) Porfírio, C.; Neto, E. A.; Odloak, D. Multi-model predictive control of an industrial C3/C4 splitter. *Control Eng. Pract.* **2003**, *11*, 765−779.

(2) Murray-Smith, R.; Johansen, T. *Multiple model approaches to nonlinear modelling and control*; Taylor & Francis: London, England, 1997.

(3) Galan, O.; Romagnoli, J. A.; Palazoglu, A. Real-time implementation of multi-linear model-based control strategies—an application to a

Table 2. Control Performance of the MPC Controllers[a]

	$\overline{T}_s$/min			T_s^{max}/min			$\overline{M}_p$		M_p^{max}	
Algorithm	ϵ = 5%	ϵ = 2%	ϵ = 1%	ϵ = 5%	ϵ = 2%	ϵ = 1%	C_A	T	C_A	T
MMPC-TS	**1.214**	**1.614**	**1.929**	**1.6**	**2.2**	**2.7**	**2.59%**	**1.07%**	**6.39%**	**5.36%**
LMPC1										
LMPC2	1.357	1.914	2.357	2.0	2.3	2.8	4.82%	3.08%	12.99%	6.08%
MMPC-HS	1.386	1.800	2.086	2.1	2.8	2.9	4.41%	2.63%	10.00%	9.22%

[a]Because LMPC1 leads to serious output oscillations in some set-points, these criteria are not applicable to LMPC1.

bench-scale pH neutralization reactor. *J. Process Control* **2004**, *14*, 571−579.

(4) Du, J.; Song, C.; Li, P. Application of gap metric to model bank determination in multilinear model approach. *J. Process Control* **2009**, *19*, 231−240.

(5) Galán, O.; Romagnoli, J. A.; PalazoÇğlu, A.; Arkun, Y. Gap metric concept and implications for multilinear model-based controller design. *Ind. Eng. Chem. Res.* **2003**, *42*, 2189−2197.

(6) Tan, W.; Marquez, H. J.; Chen, T.; Liu, J. Multimodel analysis and controller design for nonlinear processes. *Comput. Chem. Eng.* **2004**, *28*, 2667−2675.

(7) Du, J.; Song, C.; Yao, Y.; Li, P. Multilinear model decomposition of MIMO nonlinear systems and its implication for multilinear model-based control. *J. Process Control* **2013**, *23*, 271−281.

(8) Shaghaghi, D.; Fatehi, A.; Khaki-Sedigh, A. Multi-linear model set design based on the nonlinearity measure and H-gap metric. *ISA Trans.* **2017**, *68*, 1−13.

(9) 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.* **2009**, *48*, 3934−3943.

(10) Cervantes, A. L.; Agamennoni, O. E.; Figueroa, J. L. A nonlinear model predictive control system based on Wiener piecewise linear models. *J. Process Control* **2003**, *13*, 655−666.

(11) Ławryńczuk, M. Suboptimal nonlinear predictive control based on multivariable neural Hammerstein models. *Appl. Intell.* **2010**, *32*, 173−192.

(12) Fruzzetti, K.; Palazoğlu, A.; McDonald, K. Nolinear model predictive control using Hammerstein models. *J. Process Control* **1997**, *7*, 31−41.

(13) Cao, X.; Ayalew, B. *Control-oriented mimo modeling of laser-aided powder deposition processes*. American Control Conference. 2015; pp 3637−3642.

(14) Darby, M. L.; Nikolaou, M. MPC: Current practice and challenges. *Control Eng. Pract.* **2012**, *20*, 328−342.

(15) Foss, B. A.; Johansen, T. A.; Sørensen, A. V. Nonlinear predictive control using local models - applied to a batch fermentation process. *Control Eng. Pract.* **1995**, *3*, 389−396.

(16) Aufderheide, B.; Bequette, B. W. Extension of dynamic matrix control to multiple models. *Comput. Chem. Eng.* **2003**, *27*, 1079−1096.

(17) Du, J.; Johansen, T. A. A gap metric based weighting method for multimodel predictive control of MIMO nonlinear systems. *J. Process Control* **2014**, *24*, 1346−1357.

(18) Du, J.; Johansen, T. A. Integrated multimodel control of nonlinear systems based on gap metric and stability margin. *Ind. Eng. Chem. Res.* **2014**, *53*, 10206−10215.

(19) Polak, E. *Optimization: algorithms and consistent approximations*; Springer-Verlag: New York, 2012; Vol. *124*.

(20) Wang, X.; Yuan, P.; Mao, Z. The modified feature subsets ensemble applied for the mach number prediction in wind tunnel. *IEEE Trans. Aerosp. Electron. Syst.* **2016**, *52*, 863−874.

(21) Julian, P.; Desages, A.; Agamennoni, O. High-level canonical piecewise linear representation using a simplicial partition. *IEEE Trans. Circuits Syst. I, Fundam. Theory* **1999**, *46*, 463−480.

(22) Julian, P. A high level canonical piecewise linear representation: theory and applications. Ph.D. thesis, Universidad Nacional del Sur, Bahia Blanca, Argentina, 1999.

(23) Wang, L.; Young, P. C. An improved structure for model predictive control using non-minimal state space realisation. *J. Process Control* **2006**, *16*, 355−371.

(24) Zhang, R.; Wu, S.; Gao, F. State Space Model Predictive Control for Advanced Process Operation: A Review of Recent Development, New Results, and Insight. *Ind. Eng. Chem. Res.* **2017**, *56*, 5360−5394.

(25) 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* **2011**, *21*, 1183−1192.

(26) Yepes, A. G.; Vidal, A.; Malvar, J.; López, O.; Doval-Gandoy, J. Tuning method aimed at optimized settling time and overshoot for synchronous proportional-integral current control in electric machines. *IEEE Trans. Power Electron.* **2014**, *29*, 3041−3054.

(27) Aström, K. J.; Murray, R. M. *Feedback systems: an introduction for scientists and engineers*; Princeton University Press: 2010.