J Intell Robot Syst (2007) 50:297–319
DOI 10.1007/s10846-007-9166-5

Hybrid Fuzzy Modelling for Model Predictive Control

Gorazd Karer · Gašper Mušič · Igor Škrjanc ·
Borut Zupančič

Received: 19 July 2006 / Accepted: 7 June 2007 /
Published online: 18 July 2007

Abstract Model predictive control (MPC) has become an important area of research and is also an approach that has been successfully used in many industrial applications. In order to implement a MPC algorithm, a model of the process we are dealing with is needed. Due to the complex hybrid and nonlinear nature of many industrial processes, obtaining a suitable model is often a difficult task. In this paper a hybrid fuzzy modelling approach with a compact formulation is introduced. The hybrid system hierarchy is explained and the Takagi–Sugeno fuzzy formulation for the hybrid fuzzy modelling purposes is presented. An efficient method for identifying the hybrid fuzzy model is also proposed. A MPC algorithm suitable for systems with discrete inputs is treated. The benefits of the MPC algorithm employing the hybrid fuzzy model are verified on a batch-reactor simulation example: a comparison between the proposed modern intelligent (fuzzy) approach and a classic (linear) approach was made. It was established that the MPC algorithm employing the proposed hybrid fuzzy model clearly outperforms the approach where a hybrid linear model is used, which justifies the usability of the hybrid fuzzy model. The hybrid fuzzy formulation introduces a powerful model that can faithfully represent hybrid and nonlinear dynamics of systems met in industrial practice, therefore, this approach demonstrates a significant advantage for MPC resulting in a better control performance.

Keywords Fuzzy systems · Hybrid systems · Model predictive control

G. Karer (✉) · G. Mušič · I. Škrjanc · B. Zupančič
Faculty of Electrical Engineering, University of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia
e-mail: gorazd.karer@fe.uni-lj.si

1 Introduction

Dynamic systems that involve continuous and discrete states are called *hybrid systems*. Most industrial processes contain both continuous and discrete components, for instance, discrete valves, on/off switches, logical overrides, etc. The continuous dynamics are often inseparably interlaced with the discrete dynamics; therefore, a special approach to modelling and control is required. At first this topic was not treated systematically [24]. In recent years, however, hybrid systems have received a great deal of attention from the computer science and control community.

The principle of *model predictive control* (MPC) is based on forecasting the future behavior of a system at each sampling instant using the process model. The complex hybrid and nonlinear nature of many processes that are met in practice causes problems with both structure modelling and parameter identification; therefore, obtaining a model that is suitable for MPC is often a difficult task. Classic modelling and especially identification methods that originate from linear-system theory are inadequate for treating such systems. Hence, the need for special methods and formulations when dealing with hybrid systems is very clear.

MPC methods for hybrid systems employ several model formulations. Often the system is described as *mixed logical dynamical* (MLD) [3]. *Piecewise affine* (PWA) formulation [21] is also a widely used representation of hybrid systems in MPC. The PWA formulation has been proven to be equivalent to many classes of hybrid dynamical models [8], including the MLD models. The optimal control problem for discrete-time PWA models can be converted to a mixed-integer optimization problem and solved online [13]. On the other hand, in [10] the authors tackle the optimal control problem for PWA models by solving a number of multi-parametric programs offline. In such manner, it is possible to obtain a solution in the form of a PWA state feedback law that can be efficiently implemented online.

The aforementioned methods mainly consider systems with continuous inputs, despite the fact that solutions based on *(multiparametric) mixed integer linear/quadratic programming* (mp-MIQP/MILP) can be applied to systems with discrete inputs as well. However, the computational complexity increases drastically with the number of discrete states, and so these methods can become computationally too demanding.

In [15], the authors propose a MPC algorithm that uses a PWA model. The approach employs the concepts of reachable sets combined to a branch and bound method in the MPC and thus reduces the computational complexity of the QP needed to be solved in the optimization algorithm.

In case of systems with discrete inputs only, a simpler branch and bound method can be implemented [16, 17], because there is no need for the QP optimizations and only a tree of evolution can be used in order to obtain the optimal input sequence. Due to a lower computational complexity, the algorithm can be implemented in real-time on many processes.

A fuzzy model represents a powerful universal approximator for nonlinear dynamics. A hierarchical identification of a fuzzy switched system is introduced in [14]. Furthermore, two structure-selecting methods for nonlinear models with mixed discrete and continuous inputs are presented in [6]. In [18], a fuzzy control method is implemented in the low control-level for a class of hybrid systems based on hybrid automata. A nonlinear modelling approach for MPC purposes is presented in [20].

The authors introduce an analytical predictive-control-law for fuzzy systems. The modelling and identification methodology is usable for plain nonlinear systems, but not for the structurally more complex class of hybrid systems.

Nevertheless, most of the previous work related to the MPC of hybrid systems is based on (piecewise) linear and equivalent models. However, such approaches can prove unsuccessful when dealing with distinctive nonlinearities. When using a PWA model, further segmentation is required in order to suitably approximate the nonlinearity, comparing to a fuzzy model. The new segments introduce new discrete auxiliary variables in the optimization program, which causes a higher complexity, often resulting in programs that are computationally too demanding.

The basic idea of this paper is to present an efficient approach for obtaining a hybrid fuzzy model by means of identifying the unknown system. Some concepts from the previous work are extended to nonlinear hybrid systems. In the paper a formulation for a hybrid fuzzy model that is based on a hierarchical structure and can be written in a compact form is proposed. The discrete part (which is atop the hierarchy) denotes the operating modes of the system. These can be represented as a crisp coarse division of the state-space – similar to partitioning in the PWA framework. The lower level of the hierarchy, i.e. the continuous part of the hybrid fuzzy model, represents a finer subpartitioning of the state-space within the coarse partitions of the discrete level. However, the subpartitions (within a partition) are not divided by crisp borders; the borders between the subpartitions are of a fuzzy type and are defined by the appropriate membership functions. In such manner, the fuzzified subpartitions overlap and thus the nonlinear nature of a system (within an operating mode) is better approximated in the model. In other words, the PWA framework introduces crisp partitioning of the state-space, whereas in the hybrid fuzzy model the state-space within individual coarser partitions, which represent the system's operating modes, is fuzzyfied. A PWA model can actually be regarded as a special case of a fuzzy model, i.e. a fuzzy model with crisp membership functions. To put it another way, the introduction of the hybrid fuzzy model formulation extends the problem domain by generalizing the PWA approach and therefore broadening the applicability range. As known from literature [1, 11], a fuzzy model represents a more powerful universal approximator comparing to a PWA model for modelling a nonlinear system. The PWA approach needs finer partitioning of the state-space than the hybrid fuzzy approach to effectively approximate the nonlinearity, which is a significant advantage of the hybrid fuzzy model regarding the complexity of the obtained model. We suggest that the proposed hybrid fuzzy model is suitable for implementation in the MPC of nonlinear hybrid systems with discrete inputs based on a reachability analysis. By using a more accurate model for the MPC, a better control performance can be achieved, which is a significant advantage of the presented approach.

The outline of the paper is as follows. In Section 2 the structure modelling of a hybrid fuzzy model is discussed. This is followed by Section 3, which deals with the parameter estimation of the model. In Section 4 an algorithm for the MPC of systems with discrete inputs based on a reachability analysis is treated. In the following sections, a batch-reactor process is introduced. The modelling and parameter estimation of the process are tackled and the framework is verified by means of a simulation experiment. Finally, a comparison between MPC employing a hybrid linear model and a hybrid fuzzy model is made.

2 Modelling of a Hybrid Fuzzy Model

Dynamic systems are usually modelled by feeding back delayed input and output signals. In the discrete-time domain a common nonlinear model structure is the NARX (nonlinear autoregressive with exogenous inputs) model [19], which gives the mapping between the past input-output data and the predicted output.

$$\hat{y}_p(k+1) = F(y(k), y(k-1), ..., y(k-n+1), u(k), u(k-1), ..., u(k-m+1)) \quad (1)$$

Here, $y(k), y(k-1), ..., y(k-n+1)$ and $u(k), u(k-1), ..., u(k-m+1)$ denote the delayed process output and input signals, respectively. Hence, the model of the system is represented by the (nonlinear) function F.

In this paper, a special class of systems is addressed, i.e., nonlinear hybrid systems with discrete inputs. Therefore, in Eq. 1 u stands for the discrete input.

2.1 Hybrid System Hierarchy

As already mentioned, many processes met in practice demonstrate a hybrid nature, which means that the continuous dynamics are interlaced with the discrete dynamics. A special class of such systems is called switched systems, where the continuous states remain continuous even when the discrete states are changed, i.e. no jumps of the continuous state vector are allowed. In this paper we deal with hybrid systems represented by a hierarchy of discrete and continuous subsystems where the discrete part is atop the hierarchy. A discrete-time formulation is described in Eqs. 2 and 3.

$$\mathbf{x}(k+1) = \mathbf{f}_q(\mathbf{x}(k), \mathbf{u}(k)) \quad (2)$$

$$q(k) = g(\mathbf{x}(k), q(k-1), \mathbf{u}(k)) \quad (3)$$

Here, $\mathbf{x} \in \mathbb{R}^n$ is the continuous state vector, which includes all relevant system outputs y (see Eq. 1), i.e. measurable continuous states (delayed and non-delayed) that influence the state vector in the next time-step. $\mathbf{u} \in \mathbb{R}^m$ denotes the input vector. $q \in \mathbb{Q}$ (where $\mathbb{Q} = \{1, ..., s\}$) is the discrete state, which defines the switching region. Discrete states are also referred to as operating modes. There are s operating modes of the hybrid system. The hybrid states are hence described at any time-step k by the set of states $(\mathbf{x}(k), q(k))$ in the domain $\mathbb{R}^n \times \mathbb{Q}$.

The local behavior of the model described in Eq. 2 depends on the discrete state $q(k)$, which defines the current function $\mathbf{f}_q$.

Equation 3 introduces a modification of the strict Witsenhausen hybrid system formulation [25] in the sense that the discrete state $q(k)$ depends on the input vector $\mathbf{u}(k)$ as well as on the continuous state vector $\mathbf{x}(k)$ and the previous discrete state $q(k-1)$.

The continuous part of the system is generally nonlinear, therefore it can be modelled as a Takagi–Sugeno fuzzy model, as shown in the sequel.

2.2 Generalization of the Takagi–Sugeno Formulation for a Nonlinear Hybrid System

In order to approximate a nonlinear system, a fuzzy formulation can be employed. Fuzzy models can be regarded as universal approximators, which can approximate continuous functions to an arbitrary precision [5, 7].

The system dynamics can be formulated as a Takagi–Sugeno fuzzy model. In order to address nonlinear hybrid systems, we have generalized the model formulation by incorporating the discrete part of the system dynamics given in Eq. 3 in the rule base. In this instance, the rule base of the hybrid fuzzy system is represented in Eq. 4.

$$\begin{aligned}
&\mathbf{R}^{jd}: \\
&\quad \textbf{if } q(k) \text{ is } Q_d \text{ and } y(k) \text{ is } A_1^j \text{ and ... and } y(k-n+1) \text{ is } A_n^j \\
&\quad \textbf{then } \hat{y}_p(k+1) = f_{jd}(y(k), ..., y(k-n+1), u(k), ..., u(k-m+1))
\end{aligned} \tag{4}$$

$$\text{for } j = 1, ..., K \text{ and } d = 1, ..., s$$

The **if**-parts (antecedents) of the rules describe hybrid fuzzy regions in the space of the input variables of the hybrid fuzzy model. Here, $q(k) \in \{1, ..., s\}$ stands for the discrete state of the nonlinear hybrid system, i.e., its operating mode. Q_d and A_i^j represent (fuzzy) sets characterized by their crisp and fuzzy membership functions, respectively. The **then**-parts (consequences) are functions of the inputs of the hybrid fuzzy model. Here $\hat{y}_p(k+1)$ is an output variable representing the predicted output of the process in the next time step. There is one function of inputs f_{jd} defined for each rule $\mathbf{R}^{jd}$; $j = 1, ..., K$ and $d = 1, ..., s$ in the hybrid fuzzy model. In general, f_{jd} can be a nonlinear function. However, usually an affine function f_{jd} is used, as shown in Eq. 5. Here, $a_{1jd}, ..., a_{njd}, b_{1jd}, ..., b_{mjd}$ and r_{jd} denote consequent parameters, all corresponding to the rule $\mathbf{R}^{jd}$.

$$\begin{aligned}
f_{jd}(y(k), ..., y(k-n+1), u(k), ..., u(k-m+1)) = \\
= a_{1jd}\, y(k) + ... + a_{njd}\, y(k-n+1) + \\
+\, b_{1jd}\, u(k) + ... + b_{mjd}\, u(k-m+1) + r_{jd}
\end{aligned} \tag{5}$$

The number of relevant rules in the hybrid fuzzy model is $K \cdot s$. Generally speaking, K depends on the number of fuzzy membership functions for each antecedent variable $y(k), ..., y(k-n+1), u(k), ..., u(k-m+1)$. The membership functions have to cover the whole operating area of the system. What is more, the rules have to distinguish all possible combinations of the membership functions in the antecedent variable space. Hence, K is a product of the number of membership functions corresponding to each antecedent variable $y(k), y(k-1), ..., y(k-n+1), u(k), ..., u(k-m+1)$. Note that there are K fuzzy sets A_i^j as the appurtenant membership functions are the same for every rule $\mathbf{R}^{jd}$, regardless of d. On the other hand, s denotes the number of operating modes of the nonlinear hybrid system, which is also the number of crisp membership functions characterizing the sets Q_d.

The output of the hybrid fuzzy model in a compact form is given by the following equation.

$$\hat{y}_p(k+1) = \boldsymbol{\beta}(k)\ \boldsymbol{\Theta}^T(k)\ \boldsymbol{\psi}(k) \tag{6}$$

Here, $\boldsymbol{\beta}(k) = [\beta_1(k)\ \beta_2(k)\ ...\ \beta_K(k)]$ represents the normalized degrees of fulfillment for the whole set of fuzzy rules ($j = 1, ..., K$) in the current time-step k. $\beta_j(k)$, which corresponds to a set of rules $\mathbf{R}^{jd}$ for every $d = 1, ..., s$, is obtained by using a T-norm [22]. In our case it is a normalized algebraic product of the membership values $\mu_{A_1^j}(y(k))\ ...\ \mu_{A_n^j}(y(k-n+1))$ [1, 2, 22].

In Eq. 6, $\boldsymbol{\Theta}(k)$ denotes a matrix with $n+m+1$ rows and K columns, which contains the consequent fuzzyfied parameters of the hybrid fuzzy model in the current time-step k. As noted in Eq. 7, $\boldsymbol{\Theta}(k)$ is actually a function of the discrete state of the hybrid fuzzy system in the current time-step $q(k)$.

$$\boldsymbol{\Theta}(k) = \boldsymbol{\Theta}(q(k)) = \left\{ \begin{array}{lc} \boldsymbol{\Theta}_1 & \text{if} \quad q(k) = 1 \\ \vdots & \vdots \\ \boldsymbol{\Theta}_d & \text{if} \quad q(k) = d \\ \vdots & \vdots \\ \boldsymbol{\Theta}_s & \text{if} \quad q(k) = s \end{array} \right\} \tag{7}$$

The matrices $\boldsymbol{\Theta}_d$ contain the consequent fuzzyfied parameters of the hybrid fuzzy model for each operating mode ($q = d \in \{1, ..., s\}$), individually. We assume the set of matrices $\boldsymbol{\Theta}_d$ to be time-invariant.

Each matrix $\boldsymbol{\Theta}_d$ contains all the consequent fuzzyfied parameters of the hybrid fuzzy model for the set of hybrid fuzzy rules $\{\mathbf{R}^{jd}\}$, where d is fixed and $j = 1, ..., K$. A matrix $\boldsymbol{\Theta}_d$ is made up of K columns $\boldsymbol{\Theta}_{jd} = [a_{1jd}\ ...\ a_{njd}\ \ b_{1jd}\ ...\ b_{mjd}\ \ r_{jd}]^T$.

In Eq. 6, $\boldsymbol{\psi}(k) = [y(k)\ \cdots\ y(k-n+1)\ \ u(k)\ \cdots\ u(k-m+1)\ \ 1]^T$ denotes a regressor in time-step k, which contains all the relevant model inputs that are needed in f_{jd}.

In general, hybrid fuzzy models can have multiple inputs[1] and outputs. When modelling a multiple-output process, several single-output models in parallel can be used instead, without any loss of generality. Similarly, when dealing with higher-than-first-order processes ($n > 1$), it is generally more appropriate to employ several (simpler) first-order models running in parallel in place of a single nth-order model, provided it is possible to measure the relevant system states, which substitute the system outputs from the past time-steps $y(k-1),\ ...,\ y(k-n+1)$ needed to predict $\hat{y}_p(k+1)$. If such first-order models are not feasible, it is still suitable to employ several lower-than-nth-order models instead. To put it another way, it is generally reasonable to make use of all the available data measured in a single time-step. However, due to unmeasurable system states it is not always possible to carry out such an approach.

[1] If the system has several inputs, the regression vector is simply extended so as to include all the relevant model inputs.

3 Identification of the Hybrid Fuzzy Model

To identify a hybrid fuzzy system means to obtain the hybrid fuzzy model parameters $a_{1jd}, \ldots, a_{njd}, b_{1jd}, \ldots, b_{mjd}$ and r_{jd} for each rule $\mathbf{R}^{jd}$; $j = 1, ..., K$ and $d = 1, ..., s$. To put it another way, all the matrices $\mathbf{\Theta}_d$ have to be established.

The regression matrix $\mathbf{\Psi}_{jd} = [\beta_j(k_1)\ \psi^T(k_1)\ \cdots\ \beta_j(k_{Pjd})\ \psi^T(k_{Pjd})]^T$ for the rule $\mathbf{R}^{jd}$ is obtained by using the whole set of input data for the hybrid fuzzy system. Here, index k runs from k_1 to k_{Pjd}, where P_{jd} denotes the number of input-output data pairs corresponding to the rule $\mathbf{R}^{jd}$. However, only data from time-steps k that comply with the conditions $q(k) = d$ and $\beta_j(k) \geq \delta^2$ are actually used for constructing the regression matrix $\mathbf{\Psi}_{jd}$. Since the model parameters are obtained by matrix inversion , compliance with the condition on $\beta_j(k)$ is essential for obtaining suitably conditioned matrices.

The output variable of the system y is included in the output data vector $\mathbf{Y}_{jd} = [\beta^j(k_1)\ y(k_1 + 1)\ \cdots\ \beta_j(k_1)\ y(k_{Pjd} + 1)]^T$, which corresponds to the rule $\mathbf{R}^{jd}$. Again, only data from time-steps $(k + 1)$ that comply with the conditions $q(k) = d$ and $\beta_j(k) \geq \delta$ are actually used for constructing the output data vector $\mathbf{Y}_{jd}$.

The hybrid fuzzy model is established by calculating the model parameters for the whole set of rules $\mathbf{R}^{jd}$; $j = 1, ..., K$ and $d = 1, ..., s$ using the least-squares identification method $\mathbf{\Theta}_{jd} = (\mathbf{\Psi}_{jd}^T \mathbf{\Psi}_{jd})^{-1} \mathbf{\Psi}_{jd}^T \mathbf{Y}_{jd}$.

The parameters of the hybrid fuzzy model are estimated on the basis of measured input-output data using the least-squares identification method. The approach is based on decomposition of the data matrix $\mathbf{\Psi}$ into $K \cdot s$ submatrices $\mathbf{\Psi}_{jd}$. Hence, the parameters for each rule $\mathbf{R}^{jd}$ ($j = 1, ..., K$ and $d = 1, ..., s$) are calculated separately. Due to better conditioning of the submatrices $\mathbf{\Psi}_{jd}$, compared to the conditioning of the whole data matrix $\mathbf{\Psi}$, this approach leads to a better estimate of the hybrid fuzzy parameters, or to put it another way, the variances of the estimated parameters are smaller compared to the classic approach given in the literature [1, 2, 22, 23].

The model parameters can be used directly to predict the behavior of the system in MPC, while the controller has to adapt to the dynamic changes online.

4 Model Predictive Control of Systems with Discrete Inputs Based on a Reachability Analysis

Model predictive control is an approach where a model of the system is used to predict the future evolution of the system [4, 12]. The most appropriate input vector is established and applied for every time-step. Its determination is an optimization problem that is solved within a finite horizon H, i.e., for a pre-specified number of time-steps ahead. For each time-step k a sequence of optimal input vectors (8) is acquired, which minimizes the selected cost function while considering the eventual constraints of the inputs, outputs and system states. However, only the first vector of the optimal sequence is actually applied during the current time-step. In the next time-step, a new optimal sequence is determined, etc.

$$U_k^{k+H-1} = \left\{\underline{u}(k), \underline{u}(k+1), ..., \underline{u}(k+H-1)\right\} \tag{8}$$

[2] Here, δ denotes a small positive number.

4.1 Tree of Evolution

Since the proposed control algorithm is limited to systems with discrete inputs only, the possible evolution of the system over time-steps h up to a maximum prediction horizon H can be illustrated by a tree of evolution, as shown in Fig. 1 for $H = 4$ and 3 input vectors. The nodes of the tree represent reachable states, and branches connect two nodes if a transition exists between the corresponding states.

For a given root-node V_1, representing the initial state $\underline{x_i} = \underline{x}(k|k)$ and $q_i = q(k|k)$, the reachable states are computed and inserted in the tree as nodes V_i, where i indexes the nodes as they were successively computed. The notation $(k_1|k_2)$ denotes the time-step of the current node in the MPC algorithm (k_1) and the time-step, in which the algorithm started (k_2), i.e. the actual time-step in the control-process.

A cost value J_i is associated with each new node, and based on the cost value the most promising node is selected. After labelling the node as explored, new reachable states emerging from the selected node are computed. The construction of the tree of evolution continues upwards first until one of the following conditions occurs:

- The value of the cost function (see Section 4.4) at the current node is greater than the current optimal one $(J_i \geqslant J_{\text{opt}})$.[3]
- The maximum step horizon has been reached $(h = H)$.

If the first condition occurs, the node is labelled non-promising and thus eliminated from further exploration. On the other hand, if the node satisfies the second condition only, it becomes the new current optimal node $(J_{\text{opt}} = J_i)$, whereas the sequence of input vectors leading to it becomes the current optimal one.

The exploration continues from the topmost step horizon, where unexplored nodes can be found, etc., until all the nodes are explored and the optimal input vector $\underline{u_{\text{opt}}}(k)$ can be derived from the current optimal sequence. The optimal input vector is applied to the system $\underline{u_{\text{opt}}}(k)$ and the whole procedure is repeated at the next time step $k + 1$.

4.2 Complexity of the Control Problem

The complexity of the control problem with discrete inputs is primarily subject to the maximum prediction horizon (H) and the number of discrete inputs with the associated possible input values. Let us denote the first input u_1 and the associated number of discrete values m_1 (and so on). Let us assume there are l inputs. Since there are no continuous inputs, the number of combinations of input-vector values is $M_0 = m_1 \cdot ... \cdot m_l$. The worst-case complexity of the system is therefore proportional to $C_0 = M_0^H$. The complexity grows exponentially with the number of combinations of input-vector values M_0 and the maximum prediction horizon H.

Due to physical and technological constraints it is usually possible to select only a limited number (M) of combinations of input-vector values, where $M < M_0$. The worst-case complexity of the same control problem and consequently the time

[3] Before beginning the exploration of the tree of evolution, the initial value of the current optimal node is set to infinity $J_{\text{opt}} = \infty$.

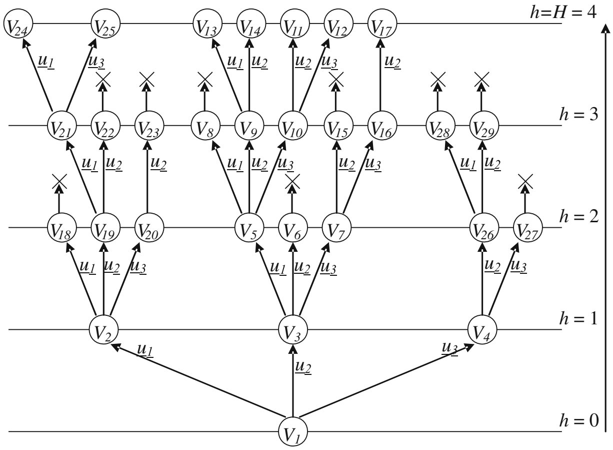


Fig. 1 Example of an explored tree of evolution. The optimal node is V_{14}, therefore the input $\underline{u}_{\text{opt}} = \underline{u}_2$ is selected and applied

needed to solve the optimization problem can thus be reduced by the factor shown in Eq. 9.

$$\frac{C_M}{C_0} = \left(\frac{M}{M_0}\right)^H \tag{9}$$

4.3 Reachability Analysis

Generally, it is not always possible to apply every input vector from the selection mentioned above, e.g. branch $\underline{u}_1$ from node V_4 (see Fig. 1). Some of the input vectors, for instance, may lead the system into undesirable states. In every time-step, such input vectors must be detected and omitted, which can be done by means of a reachability analysis. However, many of the reachable states do not lead to an optimal solution or, to put it another way, a better solution has already been found. Therefore, it is reasonable to detect and eliminate such non-promising[4] states from further exploration as early as possible, in order to reduce the complexity of the control problem. The algorithm is a kind of *branch and bound* procedure, which involves the generation of a tree of evolution that was described in Section 4.1.

The inputs, outputs and system states can be constrained for a number of reasons (either physical constraints due to the nature of the system, or constrains in connection with quality or safety in control). The physical constraints must be included in the model and are considered when building the tree of evolution by means of reachability analysis. On the other hand, the control-related constraints are usually considered in the cost function.

4.4 Cost Function

Cost-function selection has a great influence on the behaviour of the system, on the size of the fully explored tree of evolution, and hence on the computational

[4] Non-promising states are states that are definitely not leading to the optimal solution of the control problem.

complexity of the control problem. Generally, the cost function can be described as in Eq. 10.

$$\begin{aligned}
J_i &= J\left(X_k^{k+h-1}, Q_k^{k+h-1}, U_k^{k+h-2}, k, h\right)\\
&\text{for } h = 1, 2, ..., H; \text{ where}\\
X_k^{k+h-1} &= \left\{\underline{x}(k), \underline{x}(k+1), ..., \underline{x}(k+h-1)\right\};\\
Q_k^{k+h-1} &= \{q(k), q(k+1), ..., q(k+h-1)\};\\
&\text{where } \underline{x} \text{ is the vector of continuous states}\\
&\text{and } q \text{ is the discrete state of the system.}
\end{aligned} \tag{10}$$

As stated in Section 4.1, it is desirable to detect and eliminate the non-promising states from further exploration as early as possible. Since the detection is based on comparing J_i to J_{opt}, it must be ensured that no better solution than the current optimal one can be found by continuing the exploration from the eliminated node. Therefore, the cost value has to be *monotonically increasing* with the prediction-step h, as in Eq. 11.

$$\begin{aligned}
&J\left(X_k^{k+h-1}, Q_k^{k+h-1}, U_k^{k+h-2}, k, h-1\right) \leqslant\\
&\leqslant J\left(X_k^{k+h}, Q_k^{k+h-1}, U_k^{k+h-1}, k, h\right); \text{ for } h = 2, 3, ..., H.
\end{aligned} \tag{11}$$

In Eqs. 12, 13, and 14 we have proposed a universally usable form of cost function that can easily be applied to most of the systems met in practice.

$$\begin{aligned}
&J\left(X_k^{k+h}, Q_k^{k+h-1}, U_k^{k+h-1}, k, h\right) =\\
&= J\left(X_k^{k+h-1}, Q_k^{k+h-1}, U_k^{k+h-2}, k, h-1\right) +\\
&\quad + f(\underline{x}(k+h|k), q(k+h|k), \underline{u}(k+h-1|k), k, h)\\
&\text{for } h = 1, 2, ..., H
\end{aligned} \tag{12}$$

$$\begin{aligned}
&f(\underline{x}(k+h|k), q(k+h|k), \underline{u}(k+h-1|k), k, h) \geqslant 0\\
&\text{for } h = 1, 2, ..., H
\end{aligned} \tag{13}$$

$$\begin{aligned}
&J\left(X_k^k, Q_k^k, \{\}, k, 0\right) = 0\\
&\text{for } h = 0
\end{aligned} \tag{14}$$

The function f (13) is an arbitrary non-negative function that estimates the quality of control. Its value is added to the sum of the cost functions calculated along the path from the root-node to the current node in the tree of evolution. Function f basically penalizes the predicted deviations of system states from the reference trajectory (e.g., by calculating the sum of pondered squares of deviations), therefore, the cost-function value increases more rapidly along non-optimal paths in the tree of evolution. Since the function f does not have any special requirements apart from Eq. 13, it is relatively easy to determine a suitable cost function for an actual problem.

It is trivial to prove that the proposed cost function complies with condition 11.

4.5 Holding the Inputs Through a Number of Time-Steps

The maximum time of prediction reached in the control algorithm $T_{\text{pred}} = H \cdot T_{\text{S}}$ depends on two parameters: the sampling time T_{S} and the maximum prediction horizon H.

Since the sampling time T_{S} is determined by factors that in most cases can not be changed,[5] the only way to reach a longer time of prediction T_{pred} seems to be by increasing the maximum prediction horizon H. However, as stated in Section 4.2, by doing that the complexity of the control problem increases drastically.

In many cases it is possible (or even recommendable) not to change the input-vector values each sampling time. For instance, when the sampling time is relatively short, actuators could get overloaded. For this reason we have proposed an approach where the same input-vector values are kept through several (Z) time-steps or, to put it another way, the input-vector values can change only every Z time-steps.

In this case the maximum reachable time of prediction is determined as $T_{\text{pred}} = T_{\text{S}} \cdot Z \cdot H_Z$. Here, $H_Z = H/Z$ is the new maximum prediction horizon required to reach the desired maximum time of prediction T_{pred}.

The complexity of the control problem (provided T_{pred} is constant) can thus be reduced by the factor shown in Eq. 15, or rather, a longer time of prediction T_{pred} can be reached, whereas the complexity increases linearly instead of exponentially.

$$\frac{C_Z}{C_M} = \frac{(M \cdot Z)^{H_Z}}{M^H} = \left(\frac{Z}{M^{Z-1}}\right)^{H_Z} \tag{15}$$

By holding the inputs through Z time-steps several objectives can be achieved.

- Decrease the maximum prediction horizon H required to reach the desired maximum time of prediction T_{pred} and thus reduce the complexity of the control problem.
- Enable a longer maximum time of prediction T_{pred} (with the same equipment).
- Enable a shorter sampling time T_{S} while retaining the maximum time of prediction T_{pred} (on the same equipment).
- Relieve the potentially overloaded actuators.

We believe that holding the inputs through Z time steps can improve the control for a class of systems, where changing the input vector every sample time is not needed or wanted, e.g., when the fast dynamics of a stiff system is insignificant for the control performance. As mentioned, the optimal value of the parameter Z depends on the system. The higher the value of Z, the lower the maximum frequency of input changes and vice versa. Fine tuning of the parameter Z can be done by means of a simulation.

[5] E.g., the dynamics of the system, the accuracy of prediction needed and the eventual sampling time of the sensors used, etc.

Due to a longer reachable maximum time of prediction T_{pred}, we expect that increasing Z should improve the control performance. On the other hand, further increasing of Z beyond the optimum is expected to cause a deterioration of the control performance, which happens because the inputs do not change frequently enough to ensure a satisfactory control [9].

5 Batch Reactor

The usability of the control algorithm in combination with the hybrid fuzzy model has been verified on a simulation example of a real batch reactor that is situated in a pharmaceutical company and is used in the production of medicines. The goal is to control the temperature of the ingredients stirred in the reactor core so that they synthesize into the final product. In order to achieve this, the temperature has to follow the reference trajectory given in the recipe as accurately as possible.

A scheme of the batch reactor is shown in Fig. 2. The reactor's core (temperature T) is heated or cooled through the reactor's water jacket (temperature T_{w}). The heating medium in the water jacket is a mixture of fresh input water, which enters the reactor through on/off valves, and reflux water. The water is pumped into the water jacket with a constant flow ϕ. The dynamics of the system depend on the physical properties of the batch reactor, namely, the mass m and the specific heat capacity c of the ingredients in the reactor's core and in the reactor's water jacket (here, index w denotes the water jacket). λ is the thermal conductivity, S is the contact area and T_0 is the temperature of the surroundings. Furthermore, T_{in} denotes the temperature of the fresh input water, whereas $T_{\mathrm{C}} = 12°\mathrm{C}$ and $T_{\mathrm{H}} = 75°\mathrm{C}$ stand for the cool and hot input water temperatures, respectively. k_{H} and k_{C} denote the position of the on/off valves for hot and cool input water, while k_{M} is the position of the mixing valve.

The temperature of the fresh input water T_{in} depends on two inputs: the position of the on/off valves k_{H} and k_{C}. However, there are two possible operating modes of the on/off valves. In case $k_{\mathrm{C}} = 1$ and $k_{\mathrm{H}} = 0$, the input water is cool ($T_{\mathrm{in}} = T_{\mathrm{C}}$), whereas if $k_{\mathrm{C}} = 0$ and $k_{\mathrm{H}} = 1$, the input water is hot ($T_{\mathrm{in}} = T_{\mathrm{H}}$).

The ratio of fresh input water to reflux water is controlled by the third input, i.e., by the position of the mixing valve k_{M}. There are six possible ratios that can be set

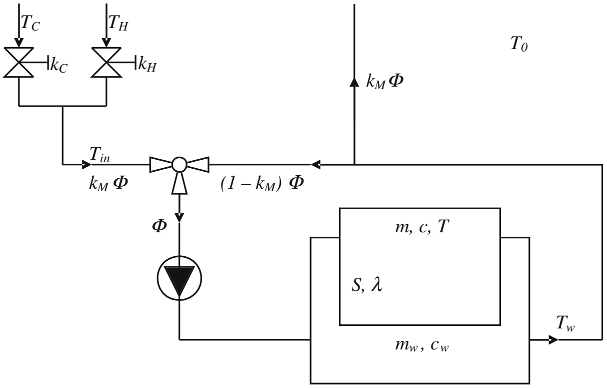


Fig. 2 Scheme of the batch reactor

by the mixing valve. The fresh input water share can be either 0, 0.01, 0.02, 0.05, 0.1 or 1.

We are therefore dealing with a multivariable system with three discrete inputs (k_{M}, k_{H} and k_{C}) and two measurable outputs (T and T_{w}).

Due to the nature of the system, the time constant of the temperature in the water jacket is obviously much shorter than the time constant of the temperature in the reactor's core. Therefore, the batch reactor is considered as a stiff system.

5.1 Modelling and Identification of the Batch Reactor

In order to apply the proposed hybrid fuzzy model structure, we first have to decompose the system to simpler MISO subsystems and establish the appropriate operating modes. On the other hand, it is also feasible to construct a model that includes the whole dynamics of the batch reactor (without first dividing it into submodels) and estimate all its parameters at once. However, since the model parameters are obtained by matrix inversion, this could result in numerical problems due to bad conditioning of the matrices used in identification. In addition, more parameters would be estimated simultaneously, which impairs the quality of identification results. By using the hybrid fuzzy model structure proposed in the sequel, we have the advantage of avoiding the possible numerical problems in identification.

The model of the batch reactor is obtained in two steps.

- The multivariable system is divided into simpler MISO subsystems.
- Each discrete-time submodel is identified individually.

The heat-flow in the reactor system can be divided as follows.

- Heat conduction between the reactor's core and the reactor's water jacket.
- Heat conduction between the reactor's water jacket and the surroundings.
- Heat convection due to an influx of fresh input water into the reactor's water jacket.
- Heat convection due to outflow of the heating medium from the reactor's water jacket.

In this manner it is possible to divide the unknown system into simpler MISO subsystems and thus apply some prior knowledge of the system in the modelling. Hence, such an identification of the unknown system is regarded as gray-box identification.

In order to carry out the identification, suitable input-output data of the batch-reactor process have to be obtained. Therefore, we generated a discrete pseudo-random signal for k_{M}[6] and an independent binary pseudo-random signal for k_{H} and k_{C} and recorded the obtained input-output data. A sampling time of $T_{\mathrm{S}} = 10$ s was used. The input signals have to cover the whole frequency and amplitude range of the system that we are interested in. This way, the identification data can characterize the complete dynamics of the system.

The time needed to acquire the data used for identification was longer than one batch cycle. However, this does not affect the dynamics of the system, because there

[6] The signal was generated using a pseudo-random mechanism that switches the possible values of k_{M} and holds a value for the duration of a pseudo-random number of time-steps.

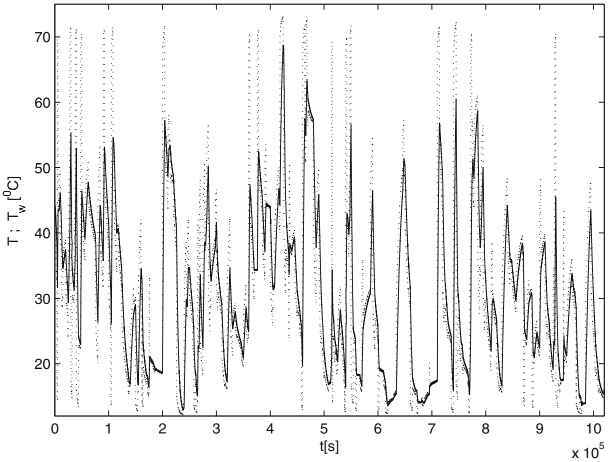


Fig. 3 Core temperature T (*solid line*) and water-jacket temperature T_w (*dotted line*)

are no endo- or exothermic reactions in the reactor. The reactor core contains granulated material that is stirred, heated and cooled in order to produce a homogenous compound.

We recorded the measurable outputs, i.e., the reactor's core temperature T and the reactor's water-jacket temperature T_w. The outputs are shown in Fig. 3. A close-up is shown in Fig. 4.

5.2 Temperature in the Reactor's Core

The temperature in the reactor's core is influenced only by the heat conduction between the reactor's core and the reactor's water jacket, which depends on the temperature difference between the reactor's water jacket T_w and the reactor's core T. Therefore, a first-order MISO submodel can be presumed, as shown in Eq. 16.

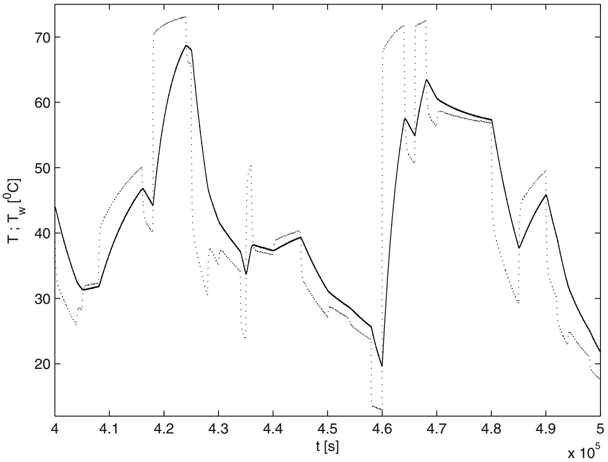


Fig. 4 Core temperature T (*solid line*) and water-jacket temperature T_w (*dotted line*)

Here, the regressor consists of the temperature in the water jacket $T_{\mathrm{w}}(k)$ and the temperature in the core $T(k)$ in the current time-step.

$$\hat{T}(k+1) = f(T_{\mathrm{w}}(k), T(k)) \tag{16}$$

Since we have surmised that the heat conduction is proportional to the temperature difference between the reactor's core and the reactor's water jacket, we can presume a linear model, as in Eq. 17.

$$\hat{T}(k+1) = \boldsymbol{\Theta}^T \left[T_{\mathrm{w}}(k) \;\; T(k) \right]^T \tag{17}$$

After conducting a least-squares identification we obtain the system parameters $\boldsymbol{\Theta}$.

$$\boldsymbol{\Theta} = [0.0033 \;\; 0.9967]^T \tag{18}$$

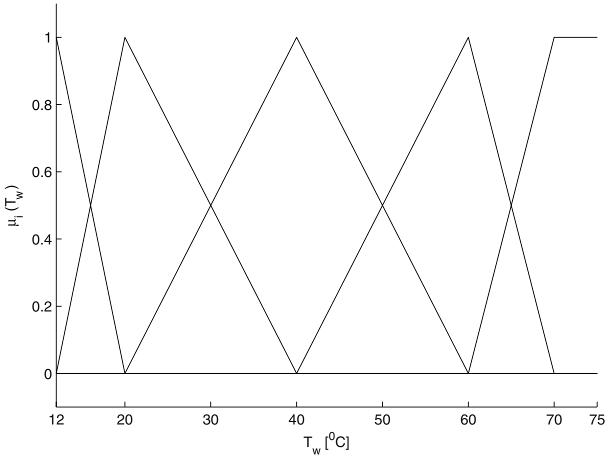
5.3 Temperature in the Reactor's Water Jacket

The temperature in the reactor water jacket T_{w} is influenced by all the heat-flow sources previously mentioned. Therefore, a MISO submodel can be presumed, as shown in Eq. 19. Here, the regressor consists of the values in the current time-step k of the temperature in the water jacket $T_{\mathrm{w}}(k)$, the temperature in the core $T(k)$, the fresh input water inflow at the mixing valve $k_{\mathrm{M}}(k)$, and the position of the cold-water and hot-water on/off valves $k_{\mathrm{C}}(k)$ and $k_{\mathrm{H}}(k)$, respectively.

$$\hat{T}_{\mathrm{w}}(k+1) = F(T_{\mathrm{w}}(k), T(k), k_{\mathrm{M}}(k), k_{\mathrm{C}}(k), k_{\mathrm{H}}(k)) \tag{19}$$

The modelling and parameter estimation of the subsystem were carried out in a similar manner to that described in Sections 2.2 and 3.

Fig. 5 Membership functions



Let us assume two operating modes of the subsystem ($s = 2$).

- The first operating mode ($q = 1$) is the case when the fresh input water is hot, i.e., $k_{\mathrm{C}}(k) = 0$ and $k_{\mathrm{H}}(k) = 1$.
- The second operating mode ($q = 2$) is the case when the fresh input water is cool, i.e., $k_{\mathrm{C}}(k) = 1$ and $k_{\mathrm{H}}(k) = 0$.

By further subdividing the MISO model in Eq. 19 we have established the discrete part of the hybrid fuzzy model, which is given in Eq. 20.

$$q(k) = q(k_{\mathrm{C}}(k), k_{\mathrm{H}}(k)) = \begin{Bmatrix} 1 & \text{if} & k_{\mathrm{C}}(k) = 0 \bigwedge k_{\mathrm{H}}(k) = 1 \\ 2 & \text{if} & k_{\mathrm{C}}(k) = 1 \bigwedge k_{\mathrm{H}}(k) = 0 \end{Bmatrix} \tag{20}$$

Next, the membership functions have to be defined. The system is fuzzyfied with regard to the temperature in the reactor's water jacket $T_{\mathrm{w}}(k)$. Simple triangular functions are used, as shown in Fig. 5.

Such a form of the membership functions ensures that the normalized degrees of fulfillment $\beta_j(T_{\mathrm{w}})$ are equal to the membership values $\mu_j(T_{\mathrm{w}})$ across the whole operating range for each rule $\mathbf{R}^{jd}$, respectively. In this case, there are five membership functions ($K = 5$), with maximums at 12, 20, 40, 60 and 70°C, so that the whole operating range is covered. In general, membership partitioning can also be done using a clustering algorithm [1].

The rule base of the hybrid fuzzy model is hence given in Eq. 21. We presume that a local system corresponding to an individual rule $\mathbf{R}^{jd}$ is affine.

$$\begin{aligned} &\mathbf{R}^{jd}: \\ &\quad \textbf{if } q(k) \text{ is } Q_d \text{ and } T_{\mathrm{w}}(k) \text{ is } A_1^j \\ &\quad \textbf{then } T_{\mathrm{w}}(k+1) = a_{1jd} T_{\mathrm{w}}(k) + a_{2jd} T(k) + b_{1jd} k_{\mathrm{M}}(k) + r_{jd} \\ &\text{for } j = 1, ..., 5 \text{ and } d = 1, 2 \end{aligned} \tag{21}$$

The output of the hybrid fuzzy model corresponding to rule $\mathbf{R}^{jd}$ can therefore be formulated as in Eq. 22.

$$\hat{T}_{\mathrm{w}}^{jd}(k+1) = \mathbf{\Theta}_{jd}^{T} \left[T_{\mathrm{w}}(k) \;\; T(k) \;\; k_{\mathrm{M}}(k) \;\; 1 \right]^{T} \tag{22}$$

After conducting a least-squares identification for each rule $\mathbf{R}^{jd}$ respectively (see Section 3), we obtain the matrices with the system parameters below for both operating modes ($\mathbf{\Theta}_1$ for $q = 1$ and $\mathbf{\Theta}_2$ for $q = 2$).

$$\mathbf{\Theta}_1 = \begin{bmatrix} 0.9453 & 0.9431 & 0.9429 & 0.9396 & 0.7910 \\ 0.0376 & 0.0458 & 0.0395 & 0.0339 & 0.0225 \\ 19.6748 & 16.7605 & 10.5969 & 3.9536 & 1.6856 \\ 0.3021 & 0.2160 & 0.5273 & 1.2701 & 12.0404 \end{bmatrix} \tag{23}$$

$$\boldsymbol{\Theta}_2 = \begin{bmatrix} 0.9803 & 0.9740 & 0.9322 & 0.9076 & 0.8945 \\ 0.0025 & 0.0153 & 0.0466 & 0.0466 & 0.0111 \\ -0.0704 & -0.6956 & -7.8013 & -12.2555 & -18.7457 \\ 0.2707 & 0.2033 & 0.5650 & 1.9179 & 5.6129 \end{bmatrix} \tag{24}$$

To sum up, the output of the model of the temperature in the reactor's water jacket is written in a compact form in Eq. 25.

$$\hat{T}_{\mathrm{w}}(k+1) = \boldsymbol{\beta}(k)\ \boldsymbol{\Theta}^T(k)\ \begin{bmatrix} T_{\mathrm{w}}(k) & T(k) & k_{\mathrm{M}}(k) & 1 \end{bmatrix}^T \tag{25}$$

6 Control

In this section, we carry out a simulation study and verify the proposed approach. In addition, we compare a modern intelligent (fuzzy) approach to a classic (linear) approach in MPC and show the advantages of the former.

In order to conduct the experiment, we first have to establish the input matrix P, which contains every allowed combination of input-vector values (see Eq. 26). Here, each column represents an input vector. The rows of the respective input vectors have the following meaning.

- The first row denotes the mixing-valve input $k_{\mathrm{M}} \in \{0, 0.01, 0.02, 0.05, 0.1, 1\}$.
- The second row denotes the cool-water on/off valve input $k_{\mathrm{C}} \in \{0, 1\}$.
- The third row denotes the hot-water on/off valve input $k_{\mathrm{H}} \in \{1, 0\}$.

$$P = \begin{bmatrix} 0 & 0.01 & 0.02 & 0.05 & 0.1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \tag{26}$$

In the last step we establish a suitable cost function (see Section 4.4). According to the prepositions in Eqs. 12, 13 and 14, a simple cost function that takes into consideration the square of deviation of the core temperature T from the reference temperature T_{ref}, is given in Eq. 27. In such a manner the control performance can be quantitatively estimated along the tree of evolution.

$$\begin{aligned} &J\left(X_k^{k+h}, Q_k^{k+h}, U_k^{k+h-1}, k, h\right) = \\ &\quad = J\left(X_k^{k+h-1}, Q_k^{k+h-1}, U_k^{k+h-2}, k, h-1\right) + \left(T(k+h) - T_{\mathrm{ref}}(k+h)\right)^2 \end{aligned} \tag{27}$$

The results of the experiment are shown in Figs. 6 and 7. The maximum prediction horizon was $H_Z = 4$ and the number of time-steps, through which the inputs are held, was $Z = 15$.

We can ascertain that the reference temperature T_{ref} was well followed by the actual core temperature T. However, there are still three obvious aspects where improvement is needed.

- The valves were moving too much.
- In several time-frames the control was carried by the on/off valves, whereas the mixing valve was relatively open. However, such behavior had been expected, for

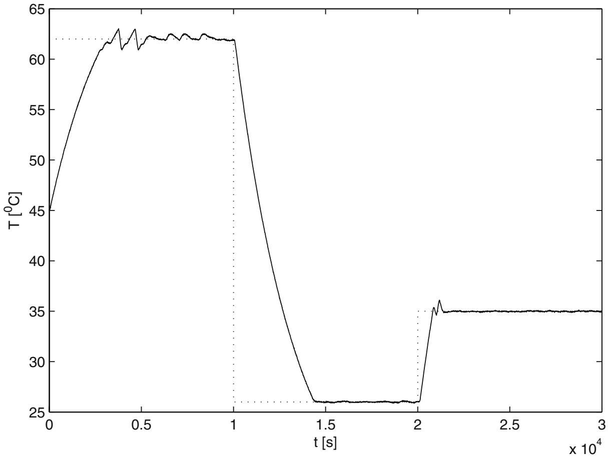


Fig. 6 Core temperature T (*solid line*) and reference temperature T_{ref} (*dotted line*)

the on/off valves can be regarded as another mixing valve, which has a greater influence on the core temperature T than the actual mixing valve.

- The total consumption of fresh input water was too high. Again, this is due to the fact that the mixing valve was fully open for too long during the experiment.

We tried to improve the problematic aspects of the control by including a sort of penalty for the movement of the valves in the cost function in Eq. 27. The modified cost function is established in Eq. 28.

$$\begin{aligned} J\left(X_k^{k+h}, Q_k^{k+h}, U_k^{k+h-1}, k, h\right) = {} & J\left(X_k^{k+h-1}, Q_k^{k+h-1}, U_k^{k+h-2}, k, h-1\right) + \\ & + w_1 \cdot (T(k+h) - T_{\mathrm{ref}}(k+h))^2 + \\ & + w_2 \cdot (k_{\mathrm{C}}(k+h) \cdot k_{\mathrm{H}}(k+h-1)) + \\ & + w_3 \cdot |k_{\mathrm{M}}(k+h) - k_{\mathrm{M}}(k+h-1)| \cdot k_{\mathrm{H}}(k+h-1) \end{aligned} \tag{28}$$

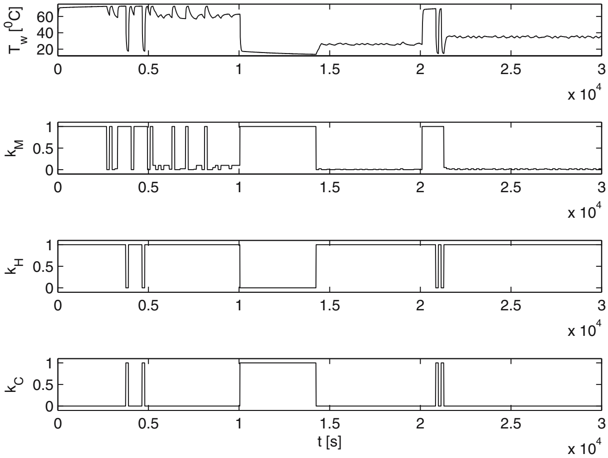


Fig. 7 Other system states

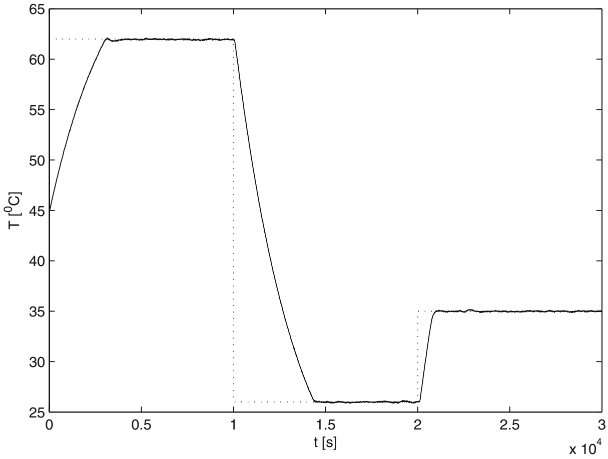


Fig. 8 Core temperature T (*solid line*) and reference temperature T_{ref} (*dotted line*)

The summands in the cost function were weighted as follows.

- The square of the deviation of the core temperature T from the reference trajectory T_{ref} was weighted according to the choice of the parameter Z, in order to enable the control performance comparison among simulations employing different Z. $w_1 = \frac{1}{Z}$.
- w_2 weights the event of changing fresh input water from hot to cool. This prevents the changes of the on/off valves when there is no negative step in the reference temperature signal T_{ref}. The weight was set as high as possible, but low enough to allow the on/off valves to change when a reference step occurs. $w_2 = 15$.
- Next, w_3 was decreased, so that the control would not be taken over exclusively by the on/off valves. $w_3 = 0.03$.

The results of the experiment are shown in Figs. 8 and 9.

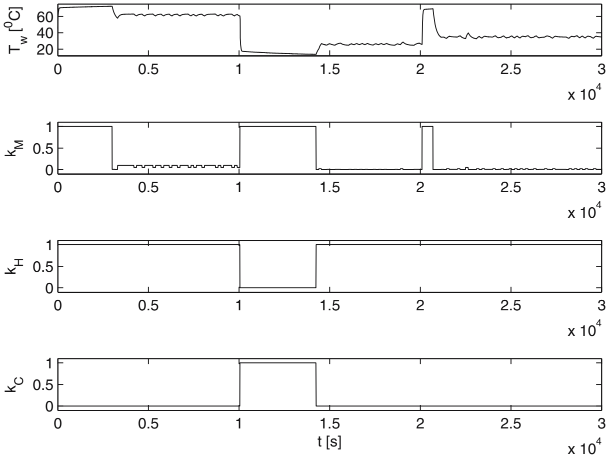


Fig. 9 Other system states

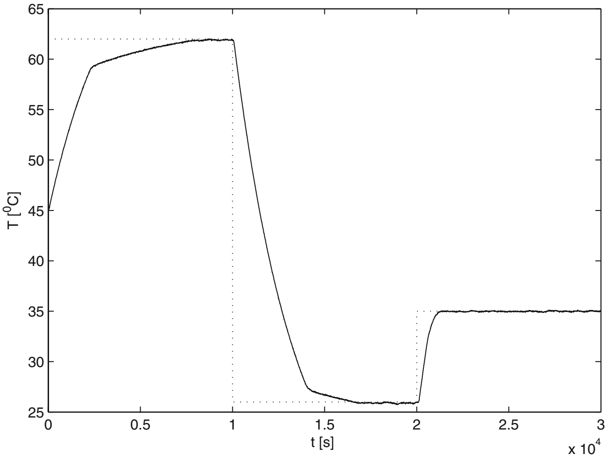


Fig. 10 Core temperature T (*solid line*) and reference temperature T_{ref} (*dotted line*)

We can ascertain that the reference temperature T_{ref} was again well followed by the actual core temperature T. What is more, by setting the cost function correctly, all three problematic aspects of the control were satisfactorily solved.

6.1 Comparison Between MPC Employing a Hybrid Fuzzy Model and a Hybrid Linear Model

In order to compare MPC employing a hybrid fuzzy model to MPC employing a hybrid linear model, we have to attain the linear model for the temperature in the reactor's water jacket (see Eq. 19). The linear model can easily be derived from the fuzzy model by linearizing it close to the center of the fuzzyfied operating range. In other words, we have used a fixed degree of fulfillment vector $\boldsymbol{\beta} = [0\ 0\ 1\ 0\ 0]$. Therefore, only one of the fuzzy regions was taken into account when establishing

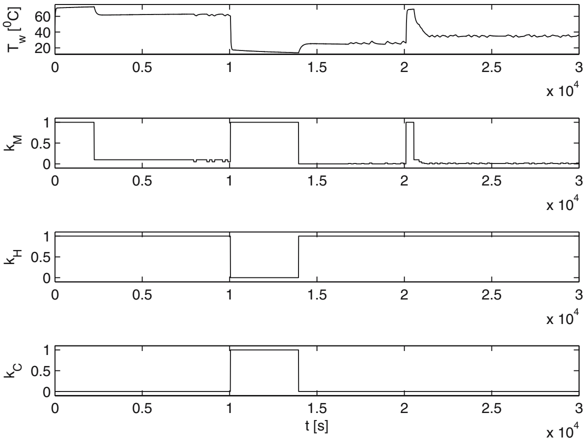


Fig. 11 Other system states

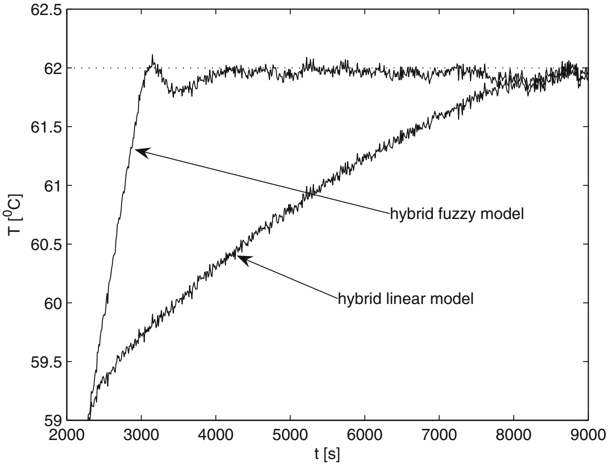


Fig. 12 Core temperature T: comparison between hybrid fuzzy model and hybrid linear model employed in MPC. $T_{\text{ref}} = 62°\text{C}$

the linear model. In this sense, the model parameters are given in Eqs. 29 and 30, whereas the output is obtained according to Eq. 31.

$$\boldsymbol{\Theta}_{1,lin} = [0.9429 \;\; 0.0395 \;\; 10.5969 \;\; 0.5273]^T \tag{29}$$

$$\boldsymbol{\Theta}_{2,lin} = [0.9322 \;\; 0.0466 \;\; -7.8013 \;\; 0.5650]^T \tag{30}$$

$$\hat{T}_{\text{w}}(k+1) = \left\{ \begin{array}{l} \boldsymbol{\Theta}^T_{1,lin} \;\; [T_{\text{w}}(k) \;\; T(k) \;\; k_{\text{M}}(k) \;\; 1]^T \text{ if } k_{\text{C}} = 0 \bigwedge k_{\text{H}} = 1 \\ \boldsymbol{\Theta}^T_{2,lin} \;\; [T_{\text{w}}(k) \;\; T(k) \;\; k_{\text{M}}(k) \;\; 1]^T \text{ if } k_{\text{C}} = 1 \bigwedge k_{\text{H}} = 0 \end{array} \right\} \tag{31}$$

The results of the experiment with the linear model employed in MPC are shown in Figs. 10 and 11. Again, the cost function in Eq. 28 was used.

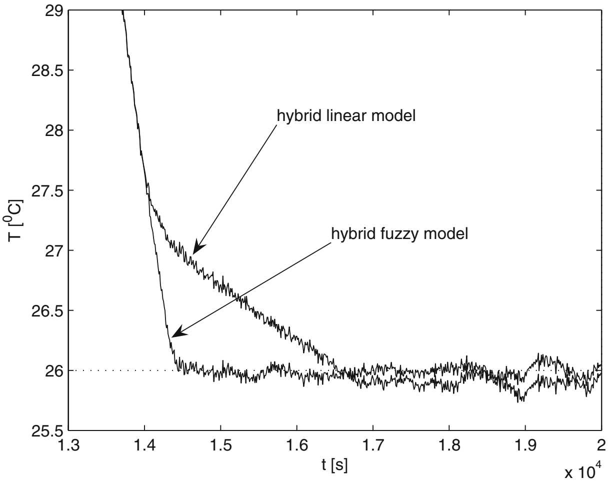


Fig. 13 Core temperature T: comparison between hybrid fuzzy model and hybrid linear model employed in MPC. $T_{\text{ref}} = 26°\text{C}$

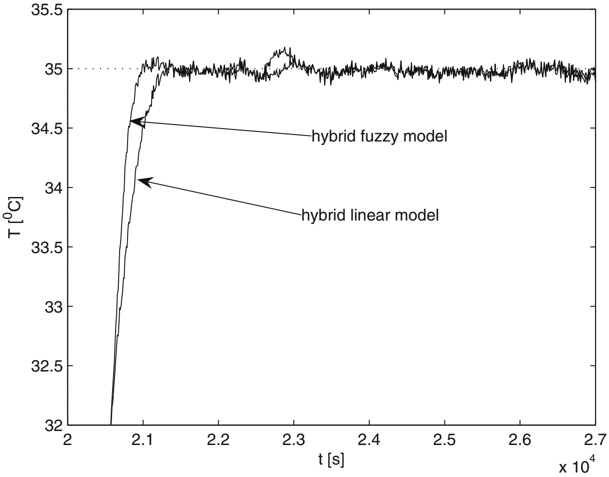


Fig. 14 Core temperature T: comparison between hybrid fuzzy model and hybrid linear model employed in MPC. $T_{\text{ref}} = 35°\text{C}$

A close-up of the relevant sections is presented in Fig. 12, where the reference temperature is $T_{\text{ref}} = 62°\text{C}$, and in Fig. 13, where the reference temperature is $T_{\text{ref}} = 26°\text{C}$. We can conclude that the MPC algorithm employing the hybrid fuzzy model clearly outperforms the case where the linear model is used.

In the third part of the experiment, where the reference temperature is $T_{\text{ref}} = 35°\text{C}$, the hybrid linear model approximates the identified system more adequately. The control performance is therefore more comparable to the case in which the hybrid fuzzy model is employed. That said, still better performance is achieved in the latter case, as can be seen in Fig. 14.

7 Conclusion

A hybrid fuzzy modelling approach for the MPC of nonlinear hybrid systems with discrete inputs based on a reachability analysis was treated. In order to implement a MPC algorithm, a suitable model of the process we are dealing with is needed. In the paper, a hybrid fuzzy modelling approach with a compact formulation was introduced. The hybrid system hierarchy was explained and the generalized Takagi–Sugeno fuzzy formulation for the hybrid fuzzy modelling purposes was presented. An efficient method for identifying the hybrid fuzzy model was also discussed.

A MPC algorithm suitable for systems with discrete inputs was treated. The benefits of the MPC algorithm employing a proposed hybrid fuzzy model was verified on a batch-reactor simulation example. The results suggest that satisfactory control can be attained, even when dealing with complex hybrid-nonlinear-stiff systems such as the batch reactor. Finally, we compared the proposed modern intelligent (fuzzy) approach to a classic (linear) approach. It was established that the MPC algorithm employing the proposed hybrid fuzzy model clearly outperforms the approach where a hybrid linear model is used.

The hybrid fuzzy formulation introduces a powerful model that can faithfully represent hybrid and nonlinear dynamics of systems met in industrial practice and can be established by means of an identification. Therefore, this approach demonstrates a significant advantage for MPC resulting in a better control performance.

References

1. Babuska, R.: Fuzzy Modeling for Control. Kluwer Academic Publishers, Norwell, MA, USA, (1998)
2. Babuska, R., Verbruggen, H.B.: An overview of fuzzy modelling for control. Control Eng. Pract. **4**(11), 1593–1606 (1996)
3. Bemporad, A., Morari, M.: Control of systems integrating logic, dynamics and constraints. Automatica **35**(3), 407–427 (1999)
4. Camacho, E.F., Bordons, C.: Model Predictive Control, Advanced Textbooks in Control and Signal Processing. Springer-Verlag, London (1998)
5. Castro, J.: Fuzzy logic controllers are universal approximators. IEEE Trans. Syst. Man Cybern. **25**, 629–635 (1995)
6. Girimonte, D., Babuska, R.: Structure for nonlinear models with mixed discrete and continuous inputs: a comparative study. In: Proc. of IEEE International Conf. on System, Man and Cybernetics, pp. 2392–2397 (2004)
7. Girosi, F., Poggio, T.: Networks and the best approximation property. Biol. Cybern. **63**, 169–176 (1990)
8. Heemels, W., Schutter, B.D., Bemporad, A.: Equivalence of hybrid dynamical models. Automatica **37**(7), 1085–1091 (2001)
9. Karer, G., Mušič, G., Zupančič B.: Predictive control of temperature in a batch reactor with discrete inputs. In: Proceedings of the 2005 IEEE International Symposium on Intelligent Control and 2005 Mediterranean Conference on Control and Automation, Limassol, pp. 855–860 (2005)
10. Kerrigan, E.C., Mayne, D.Q.: Optimal control of constrained, piecewise affine systems with bounded disturbances. In: Proc. 41st IEEE Conference on Decision and Control, Las Vegas (2002)
11. Kosko, B.: (1994) Fuzzy Systems as Universal Approximators. IEEE Trans. Comput. **43**(11), 1329–1333
12. Maciejowski, J.M.: Predictive control: with constraints. Prentice Hall, Harlow (2002)
13. Mayne, D.Q., Raković, S.: Model predictive control of constrained piecewise affine discrete-time systems. Int. J. Robust Nonlinear Control **13**(3), 261–279 (2003)
14. Palm, R., Driankov, D.: Fuzzy switched hybrid systems – Modelling and identification. In: Proc. of the 1998 IEEE/ISCI/CIRA/SAS Joint Conf., Gaithersburg MD, pp. 130–135 (1998)
15. Peña, M., Camacho, E.F., Piñón, S., Carelli, R.: Model predictive controller for piecewise affine system. In: Proc. of the IFAC World Cogress, Prague (2005)
16. Potočnik, B., Bemporad, A., Torrisi, F.D., Mušič, G., Zupančič, B.: Hybrid Modelling and Optimal Control of a Multiproduct Batch Plant. Control Eng. Pract. **12**, 1127–1137 (2004)
17. Potočnik, B., Mušič, G., Zupančič, B. : Model predictive control of discrete time hybrid systems with discrete inputs. ISA Trans. **44**(2), 199–211 (2005)
18. Qin, Y., Jia, L.-M.: Fuzzy hybrid control and its application in complex combustion processes. In: 2002 IEEE International Conference on Artificial Intelligence Systems (ICAIS'02). p. 78 (2002)
19. Sjoberg, J., Zhang, A., Ljung, L., Benveniste, A., Delyon, B., Glorennec, P.-Y., Hjalmarson, H., Juditsky, A.: Nonlonear black-box modelling in system identification: a unified overview. Automatica **31**(12), 1691–1724 (1995)
20. Škrjanc, I., Matko, D.: Fuzzy predictive functional control in the state space domain. J. Intell. Robot. Syst. **31**, 283–297 (2001)
21. Sontag, E.: 1981, Nonlinear regulation: the piecewise linear approach. IEEE Trans. Automat. Contr. **26**(2), 346–358
22. Sugeno, M., Tanaka, K.: Successive identification of a fuzzy model and its application to prediction of a complex system. Fuzzy Sets Syst. **42**, 315–334 (1991)
23. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its application to modelling and control. IEEE Trans. Syst. Man Cybern. **15**, 116–132 (1985)
24. van der Schaft, A., Schumacher, H.: (1999) An introduction to hybrid dynamical systems. Lect. Notes Control Inf. Sci. **251**, v–vii
25. Witsenhausen, H.S.: A class of hybrid-state continuous time dynamic systems. IEEE Trans. Automat. Contr. **11**(2), 161–167 (1966)