Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Fuzzy Sets and Systems 139 (2003) 59–80

FUZZY sets and systems

www.elsevier.com/locate/fss

Fuzzy model predictive control of non-linear processes using genetic algorithms

Haralambos Sarimveis*, George Bafas

School of Chemical Engineering, National Technical University of Athens, 9, Heroon Polytechniou str., Zografou Campus, Athens 15780, Greece

Received 20 September 2001; received in revised form 30 September 2002; accepted 18 October 2002

Abstract

This paper introduces a new fuzzy control technique, which belongs to the popular family of control algorithms, called Model Predictive Controllers. The method is based on a dynamic fuzzy model of the process to be controlled, which is used for predicting the future behavior of the output variables. A non-linear optimization problem is then formulated, which minimizes the difference between the model predictions and the desired trajectory over the prediction horizon and the control energy over a shorter control horizon. The problem is solved on line using a specially designed genetic algorithm, which has a number of advantages over conventional non-linear optimization techniques. The method can be used with any type of fuzzy model and is particularly useful when a direct fuzzy controller cannot be designed due to the complexity of the process and the difficulty in developing fuzzy control rules. The method is illustrated via the application to a non-linear single-input single-output reactor, where a Takagi–Sugeno model serves as a predictor of the process future behavior.
© 2002 Elsevier B.V. All rights reserved.

Keywords: Fuzzy models; Fuzzy control; Model predictive control; Genetic algorithms

1. Introduction

Model predictive control (MPC) is a family of advanced discrete-time process control algorithms, which are continuously increasing their popularity, over the last years. The philosophy of MPC techniques is based on the on-line use of an explicit process model, to predict and thereby optimize the effect of future control actions (manipulates variable changes) on the controlled variables (process outputs) over a finite horizon. MPC is the only control methodology to handle constraints in a systematic way during the design and implementation of the controller. This is the primary reason

* Corresponding author. Tel.: +30-210-772-3237; fax: +30-210-772-3138.
E-mail address: hsarimv@central.ntua.gr (H. Sarimveis).

for the success of MPC techniques in numerous industrial applications [2,4,5]. Other major attractions of MPC algorithms are the long-range predictive horizon and the fact that the control law is not fixed, but it is based on on-line optimization. Due to the above mentioned advantages, MPC algorithms can deal with:

- Multivariable systems.
- The failure of actuators (a stuck value becomes a constraint that is handled automatically and "optimally" in the algorithm).
- Non-minimum phase or dead-time systems, since they can predict the impact of current process moves to future values of process outputs.
- Model uncertainty, since a major objective in an MPC system design is robust performance. This means that the controller can be designed so that the closed loop performance specifications are met despite varying operating conditions, or what we refer to as plant/model mismatch.

The design of MPC controllers is not a trivial task. A complete model describing the effects of all the different process inputs on all the process outputs must be developed. Another difficulty is that all the control goals must be incorporated into a single objective function, which is then optimized.

The first reference on the MPC idea can be found in a late 1970s work, when Richalet et al. [27] described successful applications of Model Predictive Heuristic Control (MPHC). A number of other design techniques have been emanated from MPC, namely model algorithmic control (MAC) [20], dynamic matrix control (DMC) [9,26], internal model control (IMC) [12], linear dynamic matrix control (LDMC) [23], quadratic dynamic matrix control (QDMC) [13] and generalized predictive Control (GPC) [7,8].

All the above MPC algorithms assume a linear model of the plant. However, in many industrial cases, the dynamics of the system are highly non-linear, so that a linear MPC controller will not give rise to a satisfactory dynamic response of the plant, unless the operating conditions are very close to the steady state around which the model is linearized. In order to apply the MPC philosophy to these situations, researchers developed a new family of MPC algorithms, called Non-linear MPC (NMPC) methodologies, which predict the future behavior of the plant based on a non-linear model of the process [3].

The introduction of non-linearities in the MPC formulation enhanced the popularity and applications of the family of MPC systems, but introduced two new problems. The first concerns the model development, which is much more difficult compared to the linear case, where the most popular methodology produces impulse response models by watching the behavior of the process to simple step changes on the manipulated variables. In non-linear systems, the models are usually developed by one of the following methodologies:

(a) Generation of a number of impulse response models in different regions of the input space defined by the input variables. Practically this requires a number of step tests on the input variables starting from different steady states of the process.
(b) Development of non-linear models based on fundamental equations like mass and energy balances, thermodynamics, transport phenomena, etc. However, in the majority of applications, the dynamic behavior of the process is so complicated, that it cannot easily be approximated by such first principle models.

(c) Development of black-box models, which are based only on process input–output dynamical data. Ideally, special tests, should be performed, which perturbate the process inputs in order to collect high quality data. If this is not possible, a moderate black-box model can be developed based on process data, collected during regular operation of the process. In many cases, fuzzy models are preferred compared to other non-linear black-box models, since they are simpler in structure, easier to develop and can also incorporate human logic.

The second important problem in NMPC is that the optimization problem which needs to be solved on-line is no longer a linear problem with an explicit solution, but a complicated non-linear optimization problem which requires a tremendous computational effort. In many cases this cannot be completed on-time even by the fast computing systems of our days.

In this paper we propose a new Fuzzy MPC (FMPC) methodology, which is based on a dynamic non-linear fuzzy model of the plant and belongs to the family of indirect fuzzy control methodologies [35]. A number of methods to incorporate fuzzy models into the MPC framework have been reported in the literature: Kim et al. [18] treated the fuzzy logic model as a collection of piecewise linear models, while Huang et al. [16] described the process by a fuzzy convolution model, consisting of a number of quasi linear fuzzy implications. Fiscer et al. [10] utilized a Takagi–Sugeno model with Gaussian membership function, which solved the on-line optimization problem using an algorithm where the grid search, Hooke–Jeeves search and Newtons method are combined. Sousa et al. [31] also used a dynamic Takagi–Sugeno model for the prediction of the process behavior in the future. A model inversion combined with the branch and bound algorithm served as the basis for the solution of the optimization problem. Finally, Vucovic [34] and de Oliveira and Lemos [24] utilized relational fuzzy structures for representing the dynamics of the process. In [34], the control algorithm is selected by trying a set of predefined control actions on the model, while in [24] two control schemes are proposed: an incremental controller and an optimized relational controller. A common characteristic of all the above methodologies is that they require a specific type of fuzzy model. Moreover, the algorithms are not able to handle process constraints on the input variables, which is one of the most important advantages of the MPC family of controllers.

The proposed FMPC method can be used for any type of fuzzy logic model and takes care of the constraints on the input variables and input moves, exploiting in that way all the advantages of the MPC framework. The proposed method formulates a dynamic non-linear optimization problem, where the objective function consists of two terms: the differences between the fuzzy model predictions and the desired output trajectory over a prediction horizon, and the control energy over a control horizon. However, due to the peculiarity of fuzzy models, conventional optimization techniques cannot be easily applied. The on-line optimization problem is solved using a specially developed genetic algorithm, which guarantees the feasibility of all the generated potential solutions.

Genetic algorithms are model machine learning methodologies, which derive their behavior from a metaphor of the processes of evolution in nature and are able to overcome complex non-linear optimization tasks like non-convex problems, non-continuous objective functions, etc. [15,21]. They are based on an initial random population of solutions and an iterative procedure, which improves the characteristics of the population and produces solutions that are closer to the global optimum. This is achieved by applying a number of genetic operators to the population, in order to produce the next generation of solutions. GAs have been used successfully in combinations with fuzzy systems [14,29]. Particularly in fuzzy control, GAs have been utilized extensively to tune the fuzzy logic controllers and acquire the fuzzy rules [17,19,30,32,33]. A more comprehensive list of papers referring to fuzzy control using GAs can be found in [11]. However, most of these papers are utilizing GAs to tune the controllers off-line. Regarding on-line MPC control schemes very few references can be found in the literature that utilize GAs for the solution of the optimization problem [22,25] and according to our knowledge none of them utilizes fuzzy models for the prediction of the process behavior. The main drawback is that on-line GAs need a certain number of generations to converge, so that they cannot be applied to systems with short time constants and sample times. However, there is a considerable scope for the development of further and more powerful algorithms, since the high computational speed and the improved performance of these new algorithms will increase the viability of on-line GAs towards systems with shorter time constants [11]. In the present paper, the proposed on-line GA was utilized for the solution of the non-linear MPC optimization problem that was formulated for a single-input single-output (SISO) system and had a good performance although the sample time was only 1 min.

The rest of the paper is organized as follows: In Section 2 a brief description of dynamic fuzzy modeling is presented. Section 3 formulates the complete optimization problem, which needs to be solved in each time step to determine the next control move, provided that a fuzzy model is available for the prediction of the process behavior over a prediction horizon. In the same section we develop the genetic algorithm that is utilized to solve the aforementioned optimization problem. Section 4 describes the fuzzy identification algorithm, which is based on the subtractive clustering technique. The algorithm is used in this paper for developing dynamic Takagi–Sugeno fuzzy models. In Section 5 the proposed FMPC methodology is applied to a SISO chemical reactor and the results are discussed and compared to those produced by application of the LDMC method. The paper ends with the concluding remarks.

2. Dynamic fuzzy modeling

Fuzzy models is the name of systems which use some concepts from fuzzy logic (fuzzy sets, linguistic variables, etc.). An important difference compared to other modeling techniques is that they can easily incorporate knowledge which is provided by human experts and they do not depend only on numerical data collected from the process. Engineering fuzzy models should have real-valued input and output variables and are classified into two main categories: Mamdani models and Takagi–Sugeno models. The main part of a fuzzy model is a collection of fuzzy rules which have the following form:

$$\begin{aligned} R^l:\quad & \text{IF } x_1 \text{ is } A_1^l \text{ and } x_2 \text{ is } A_2^l \text{ and} \ldots \text{and } x_n \text{ is } A_n^l \\ & \qquad \text{THEN } y_1 \text{ is } B_1^l \text{ and } y_2 \text{ is } B_2^l \text{ and} \ldots \text{and } y_m \text{ is } B_m^l \end{aligned} \tag{1}$$

for the Mamdani models and

$$\begin{aligned} R^l:\quad & \text{IF } x_1 \text{ is } A_1^l \text{ and } x_2 \text{ is } A_2^l \text{ and} \ldots \text{and } x_n \text{ is } A_n^l \\ & \qquad \text{THEN } y_i = b_{i,0}^l + b_{i,1}^l x_1 + b_{i,2}^l x_2 + \cdots + b_{i,n}^l x_n, \quad 1 \leqslant i \leqslant m \end{aligned} \tag{2}$$

for the Takagi–Sugeno models.

In the above equations $x_1, x_2, \ldots, x_n$ are the input variables $A_1^l, A_2^l, \ldots, A_n^l$ are fuzzy sets defined on the respective universes of discourse, $y_1, y_2, \ldots, y_m$ are the output variables, $B_1^l, B_2^l, \ldots, B_m^l$ are fuzzy sets defined for the output variables and $b_{i,0}^l, b_{i,1}^l, b_{i,2}^l x_2, \ldots, b_{i,n}^l$, $1 \leqslant i \leqslant m$ are real numbers. The collection of fuzzy rules is called fuzzy inference of the system. The system is completed by the fuzzification and defuzzification procedures which fuzzify the input variables and defuzzify the output variables. The main difference between the above two categories can be found in the consequent parts of the fuzzy rules. In the Mamdani models the THEN part is fuzzy, while in the Takagi–Sugeno case the consequent part is crisp and is expressed as a linear combination of the input variables.

In a dynamic fuzzy model, at time point k, past values of the process input and output variables constitute the input variables to the system. We will use only input variables, provided that enough past values will be considered in the model. Given a process with *ni* process inputs and *no* process outputs, Eqs. (1) and (2) can be rewritten in a matrix form as

$$\begin{aligned} R^l\colon\quad & \text{IF } \mathbf{u}(k-1) \text{ is } \mathbf{A}_1^l \text{ and } \mathbf{u}(k-2) \text{ is } \mathbf{A}_2^l \text{ and} \ldots \text{and } \mathbf{u}(k-N) \text{ is } \mathbf{A}_N^l \\ & \quad \text{THEN } \mathbf{y}(k) \text{ is } \mathbf{B}^l \end{aligned} \tag{3}$$

for the Mamdani models and

$$\begin{aligned} R^l\colon\quad & \text{IF } \mathbf{u}(k-1) \text{ is } \mathbf{A}_1^l \text{ and } \mathbf{u}(k-2) \text{ is } \mathbf{A}_2^l \text{ and} \ldots \text{and } \mathbf{u}(k-N) \text{ is } \mathbf{A}_N^l \\ & \quad \text{THEN } \mathbf{y}(k) = \mathbf{h}^l + \mathbf{H}_1^l\mathbf{u}(k-1) + \mathbf{H}_2^l\mathbf{u}(k-2) + \cdots + \mathbf{H}_N^l\mathbf{u}(k-N) \end{aligned} \tag{4}$$

for the Takagi–Sugeno models.

In the above equations $\mathbf{u}(k-1), \mathbf{u}(k-2), \ldots, \mathbf{u}(k-N)$ are vectors of dimension *ni*, while $\mathbf{y}(k)$ is a vector of dimension *no*. Moreover, $\mathbf{A}_1^l$ is fuzzy set vector of length *ni*, while $\mathbf{B}^l$ is a fuzzy set vector of dimension *no*. In Eq. (4) $\mathbf{h}^l$ is a real vector of length *no* and $\mathbf{H}_1^l, \mathbf{H}_2^l, \ldots, \mathbf{H}_N^l$ are matrices with dimensions $(no \times ni)$. The input vector to the model at time point k consists of N past values of all process input variables and can be presented as follows:

$$\mathbf{x}(k) = \begin{bmatrix} \mathbf{u}(k-1) \\ \mathbf{u}(k-2) \\ \vdots \\ \mathbf{u}(k-N) \end{bmatrix}^{\mathrm{T}}. \tag{5}$$

Obviously the length of the vector $\mathbf{x}(k)$ is $(ni \cdot N)$. Using a defuzzification procedure a crisp value can be estimated for each one of the process output variables at time point k. This value will be a non-linear function of $\mathbf{x}(k)$ and can be represented by the following equation:

$$\hat{y}_i(k) = f_i(\mathbf{x}(k)), \quad 1 \leqslant i \leqslant no. \tag{6}$$

The above description gives only the basic characteristics of a dynamic fuzzy logic model. Numerous algorithms have been proposed for the definition of fuzzy sets, the derivation of fuzzy rules and the development of fuzzification and defuzzification procedures. In this paper we will use the subtractive clustering methodology for generating Takagi–Sugeno type dynamical models. The method will be described in the sequel.

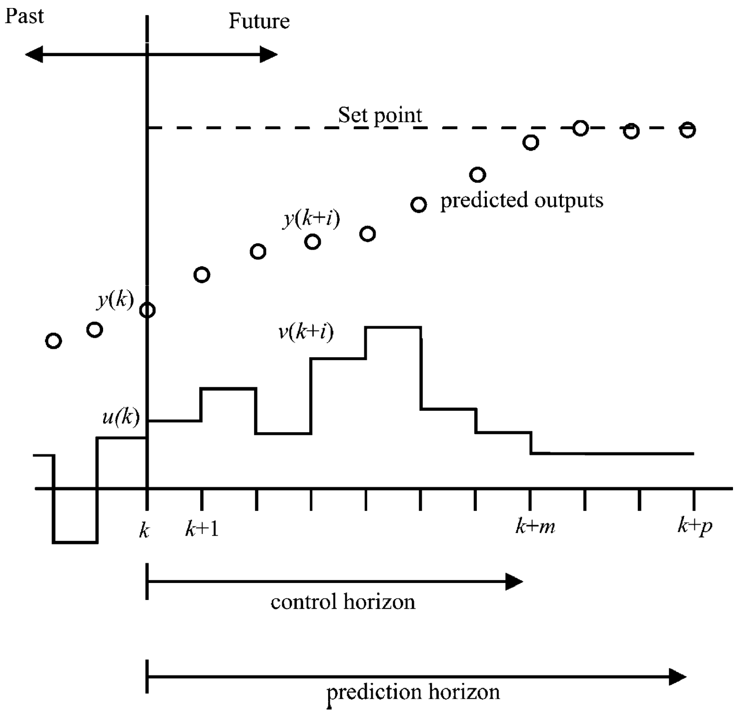


Fig. 1. A schematic view of the MPC methodology.

3. FMPC–Solving the problem using genetic algorithms

The main idea behind MPC-type controllers is illustrated in Fig. 1 for a SISO system. At sampling time k, a set of m future manipulated variable moves (control horizon) are selected, so that the predicted response over a finite horizon p (prediction horizon) has certain desirable characteristics. This is achieved by minimizing an objective function based on the deviation of the future controlled variables from a desired trajectory over the prediction horizon p and the control energy over the control horizon m. The MPC optimization is performed for a sequence of hypothetical future control moves over the control horizon and only the first move is implemented. The problem is solved again at time $k+1$ with the measured output $y(k+1)$ as the new starting point. Model uncertainty and unmeasured process disturbances are handled by calculating an additive disturbance as the difference between the process measurement and the model prediction at the current time step. For the measured disturbances it is assumed that the future values will be equal to the current values.

The proposed FMPC formulation is based on a fuzzy model, which can be used to predict the current and future values of the output variables, as described in Section 2. The model can be written in the following general vector form:

$$\hat{\mathbf{y}}(k \mid k) = \mathbf{F}(\mathbf{u}(k-1), \mathbf{u}(k-2), \ldots, \mathbf{u}(k-N))$$

$$= \begin{bmatrix} f_1(\mathbf{u}(k-1), \mathbf{u}(k-2), \ldots, \mathbf{u}(k-N)) \\ f_2(\mathbf{u}(k-1), \mathbf{u}(k-2), \ldots, \mathbf{u}(k-N)) \\ \vdots \\ f_{no}(\mathbf{u}(k-1), \mathbf{u}(k-2), \ldots, \mathbf{u}(k-N)) \end{bmatrix}, \tag{7}$$

where $\hat{\mathbf{y}}(k\,|\,k) \in \Re^{no}$ is the vector containing the fuzzy model estimations at time point k, for the current values of the output variables.

Assuming that such a fuzzy model is available, the future control moves are computed by solving an on-line optimization problem, where the objective function $J(k)$ can be expressed as a summation of vector 1-norms or 2-norms:

$$J(k) = \sum_{i=1}^{p} \|\Theta(\hat{\mathbf{y}}(k+i\,|\,k) - \mathbf{y}^{\mathrm{sp}})\|_1 + \sum_{i=0}^{m} \|\mathbf{R}_i \Delta \mathbf{v}(k+i)\|_1, \tag{8}$$

$$J(k) = \sum_{i=1}^{p} \|\Theta(\hat{\mathbf{y}}(k+i\,|\,k) - \mathbf{y}^{\mathrm{sp}})\|_2^2 + \sum_{i=0}^{m} \|\mathbf{R}_i^{1/2} \Delta \mathbf{v}(k+i)\|_2^2. \tag{9}$$

The full minimization problem, which is solved at each sampling point k, consists of the objective function, subject to the following constraints:

Model based output prediction:

$$\hat{\mathbf{y}}(k+i\,|\,k) = \mathbf{d}(k\,|\,k) + \mathbf{F}(\mathbf{v}(k+i-1), \mathbf{v}(k+i-2), \ldots, \mathbf{v}(k), \mathbf{u}(k-1), \mathbf{u}(k-2), \ldots, \mathbf{u}(k+i-N)). \tag{10}$$

Disturbance estimation:

$$\mathbf{d}(k\,|\,k) = \mathbf{y}(k) - \hat{\mathbf{y}}(k\,|\,k) = \mathbf{y}(k) - \mathbf{F}(\mathbf{u}(k-1), \mathbf{u}(k-2), \ldots, \mathbf{u}(k-N)). \tag{11}$$

Input move constraints:

$$-\Delta u_{j_{\max}} \leqslant \Delta u_j(k+i) \leqslant \Delta u_{j_{\max}}, \quad 1 \leqslant j \leqslant ni, \; 0 \leqslant i \leqslant m. \tag{12}$$

Input constraints:

$$u_{j_{\min}} \leqslant u_j(k+i) \leqslant u_{j_{\max}}, \quad 1 \leqslant j \leqslant ni, \; 0 \leqslant i \leqslant m \tag{13}$$

In the above equations the following notation is used:

- Δ is the backward difference operator, i.e. $\Delta x(k) = x(k) - x(k-1)$;
- $\hat{\mathbf{y}}(k+i\,|\,k) \in \Re^{no}$ is the vector containing the fuzzy model predictions generated at time point k, for the values of the output variables at time point $k+i$;
- $\mathbf{y}^{\mathrm{sp}} \in \Re^{no}$ is the output setpoint vector;
- $\mathbf{d}(k\,|\,k) \in \Re^{no}$ is the current disturbance vector defined as the difference between the actual output values and the fuzzy model predictions;

- $\mathbf{u}(k-1),\ldots,\mathbf{u}(k-N)\in\Re^{ni}$ are vectors containing the measured values of the input variables at time points $k-1,\ldots,k-N$;
- $\mathbf{v}(k),\ldots,\mathbf{v}(k+m)\in\Re^{ni}$ and $\Delta\mathbf{v}(k),\ldots,\Delta\mathbf{v}(k+m)\in\Re^{ni}$ are free variables (the values of the current and future input and input move vectors);
- $u_{j_{\min}}$ and $u_{j_{\max}}$ are the upper and lower bounds on $u_j(k)$, respectively, $1\leqslant j\leqslant ni$;
- $\Delta u_{j_{\max}}$ is the upper bound on the absolute value $|\Delta u_j(k)|$, $1\leqslant j\leqslant ni$;
- $\boldsymbol{\Theta}\in\Re^{no\times no}$ is the diagonal weight matrix for the output deviation term in $J(k)$;
- $\mathbf{R}_j\in\Re^{ni\times ni}$, $0\leqslant j\leqslant m$ are the input move suppression diagonal weight matrices in $J(k)$.

The variables $\Delta\mathbf{v}(k),\Delta\mathbf{v}(k+i),\ldots,\Delta\mathbf{v}(k+m)$ used in the optimization problems are just dummy variables. Only the first input move $\Delta\mathbf{v}(k)$ is implemented so that $\Delta\mathbf{u}(k)=\Delta\mathbf{v}(k)$ or $\mathbf{u}(k)=\mathbf{u}(k-1)+\Delta\mathbf{v}(k)$ and the problem is solved again at time point $k+1$.

Remark 4.1. In Eq. (10), the current disturbance is added to all future predictions, under the assumption that all unmeasured disturbances will remain unaltered during the prediction horizon.

The fuzzy MPC architecture defines a complicated non-linear optimization problem, which cannot be solved easily by classical non-linear optimization techniques. Considering that the optimization problem must be continuously solved on-line, it is obvious that we need to seek for an alternative approach. It seems that the formulation of the problem suits perfectly to the probabilistic optimization techniques, which are called genetic algorithms. As mentioned in Section 1, genetic algorithms are based on a genetic representation of possible solutions of an optimization problem and utilize search methods, which try to model some natural phenomena, like the genetic inheritance.

In the rest of this section we will describe the genetic algorithm, which was especially developed for the solution of the aforementioned optimization problem. The algorithm starts with an initial population of chromosomes, which represent possible solutions of the optimization problem. For each chromosome the objective function is computed. New generations are produced by the genetic operators that are known as crossover and mutation. The algorithm stops after the maximum allowed time has passed. In the following description, a chromosome will be represented by s^l and will have the following structure:

$$s^l=\begin{bmatrix}\mathbf{v}^l(k)\\ \mathbf{v}^l(k+1)\\ \vdots\\ \mathbf{v}^l(k+m)\end{bmatrix}^{\mathrm{T}}. \tag{14}$$

Assuming we have selected the number of chromosomes L, which will constitute the initial population, the probability of crossover p_{c} and the probability of mutation p_{m}, the algorithm can be described as follows:

Step 1: Set the number of iterations $iter=1$. Choose an initial population consisting of L chromosomes of the above from (Eq. 14). The values are chosen randomly, but they should satisfy both input and input move constraints (12)–(13). For this purpose and for each input variable $j = 1, \dots, ni$, we are using a simple procedure, which is described next:

(a) Read the measured values of the input variables at the previous time point $k-1$, which have already been implemented.
(b) Select the current input value using the following equations:

$$v_j^l(k) = u_j(k-1) + r \cdot \Delta u_{j\max}, \tag{15}$$

$$\text{If } v_j^l(k) \geqslant u_{j\max} \text{ set } v_j^l(k) = u_{j\max}. \tag{16}$$

$$\text{If } v_j^l(k) \leqslant u_{j\min} \text{ set } v_j^l(k) = u_{j\min}. \tag{17}$$

(c) Select the rest of the input moves using:

$$v_j^l(k+i) = v_j^l(k+i-1) + r \cdot \Delta u_{j\max} \quad 1 \leqslant i \leqslant m. \tag{18}$$

$$\text{If } v_j^l(k+i) \geqslant u_{j\max} \text{ set } v_j^l(k+i) = u_{j\max} \quad 1 \leqslant i \leqslant m. \tag{19}$$

$$\text{If } v_j^l(k+i) \leqslant u_{j\min} \text{ set } v_j^l(k+i) = u_{j\min} \quad 1 \leqslant i \leqslant m. \tag{20}$$

In the above equations r is a random number between -1 and 1. A new random number r is generated each time Eqs. (15) or (18) are used.

Step 2: Evaluate the objective function (8) or (9) for all the chosen chromosomes. Then invert the objective function values and find the total fitness of the population as follows:

$$V = \sum_{l=1}^{L} \frac{1}{J^l(k)}. \tag{21}$$

Step 3: Calculate the probability p_l of selection of each chromosome using the following equation:

$$p_l = \frac{(1/J^l(k))}{V} \quad 1 \leqslant l \leqslant L \tag{22}$$

Step 4: Calculate the cumulative probability q_l for each chromosome:

$$q_l = \sum_{n=1}^{l} p_n \quad 1 \leqslant l \leqslant L \tag{23}$$

Step 5: For $l = 1, \dots, L$ generate a random number r between 0 and 1. Select the chromosome for which $q_i - 1 < r \leqslant q_i$. At this point of the algorithm a new population of chromosomes has been generated. The new population probably contains multiple copies of the candidates, which gave low values of the objective function and does not include any copy of the worst chromosomes. The new population will undergo the crossover and the mutation operation, which are described next.

Step 6: Apply the crossover operation as follows: For each chromosome s^l generate a random number r in [0,1]. If r is lower than p_c, this particular chromosome will undergo the process of crossover, otherwise it will remain unchanged. Mate selected chromosomes and for each selected pair

$$s^{l} = \left[\begin{array}{c} \mathbf{v}^{l}(k) \\ \mathbf{v}^{l}(k+1) \\ \vdots \\ \mathbf{v}^{l}(k+z-1) \\ \hdashline \mathbf{v}^{l}(k+z) \\ \vdots \\ \mathbf{v}^{l}(k+m) \end{array}\right] \qquad s^{l+1} = \left[\begin{array}{c} \mathbf{v}^{l+1}(k) \\ \mathbf{v}^{l+1}(k+1) \\ \vdots \\ \mathbf{v}^{l+1}(k+z-1) \\ \hdashline \mathbf{v}^{l+1}(k+z) \\ \vdots \\ \mathbf{v}^{l+1}(k+m) \end{array}\right]$$

$$\downarrow$$

$$s^{l_{new}} = \left[\begin{array}{c} \mathbf{v}^{l}(k) \\ \mathbf{v}^{l}(k+1) \\ \vdots \\ \mathbf{v}^{l}(k+z-1) \\ \hdashline \mathbf{v}^{l+1}(k+z) \\ \vdots \\ \mathbf{v}^{l+1}(k+m) \end{array}\right] \qquad s^{l+1_{new}} = \left[\begin{array}{c} \mathbf{v}^{l+1}(k) \\ \mathbf{v}^{l+1}(k+1) \\ \vdots \\ \mathbf{v}^{l+1}(k+z-1) \\ \hdashline \mathbf{v}^{l}(k+z) \\ \vdots \\ \mathbf{v}^{l}(k+m) \end{array}\right]$$

Fig. 2. The crossover genetic operator.

generate a random integer number z between 0 and m. The crossing point is the position indicated by the random number. Two new chromosomes are produced by interchanging all the members of the parents following the crossing point. Graphically, the crossover operation can be represented as shown in Fig. 2, assuming that the crossover operation is applied to the parent chromosomes s^{l} and s^{l+1}:

The above operation might produce infeasible offspring if the input values at the crosspoint do not satisfy the input move constraints. This situation is avoided by the following correction mechanism for each input variable j, which modifies the values of the input parameters after the crossing position, so that the input move constraints are satisfied:
If

$$v_j^{l+1}(k+z) - v_j^{l}(k+z-1) > \Delta u_{j\max}. \tag{24}$$

Then

$$\varDelta = v_j^{l+1}(k+z) - v_j^{l}(k+z-1) - \Delta u_{j\max}, \tag{25}$$

$$v_j^{l+1}(k+z+i) = v_j^{l+1}(k+z+i) - \varDelta, \quad 0 \leqslant i \leqslant m-z. \tag{26}$$

If

$$v_j^{l+1}(k+z) - v_j^{l}(k+z-1) < -\Delta u_{j\max} \tag{27}$$

Then

$$\Delta = v_j^{l+1}(k+z-1) - v_j^l(k+z) - \Delta u_{j\max}, \tag{28}$$

$$v_j^{l+1}(k+z+i) = v_j^{l+1}(k+z+i) + \Delta, \quad 0 \leqslant i \leqslant m-z. \tag{29}$$

The above set of equations concerns one of the produced chromosomes, $s^{l,\text{new}}$. A similar set can be written for the chromosome $s^{l+1,\text{new}}$.

Step 7: Apply the mutation operation as follows: For each member of each chromosome s^l, $v_j^l(k+i)$, generate a random number r in [0,1]. If r is lower than p_m, this particular member of the chromosome will undergo the process of crossover, otherwise it will remain unchanged. For the selected members define upper and lower bounds as follows:

$$\left.\begin{aligned} b_u &= \min(\Delta u_{j\max} + u_j(k-1),\ \Delta u_{j\max} + v_j^l(k+i+1),\ u_{j\max}) \\ b_l &= \max(-\Delta u_{j\max} + u_j(k-1),\ -\Delta u_{j\max} + v_j^l(k+i+1),\ u_{j\min}) \end{aligned}\right\} \text{ if } i=0, \tag{30}$$

$$\left.\begin{aligned} b_u &= \min(\Delta u_{j\max} + v_j^l(k+i-1), \Delta u_{j\max} + v_j^l(k+i+1),\ u_{j\max}) \\ b_l &= \max(-\Delta u_{j\max} + v_j^l(k+i-1), -\Delta u_{j\max} + v_j^l(k+i+1),\ u_{j\min}) \end{aligned}\right\} \text{ if } 0<i<m, \tag{31}$$

$$\left.\begin{aligned} b_u &= \min(\Delta u_{j\max} + v_j^l(k+i-1),\ u_{j\max}) \\ b_l &= \max(-\Delta u_{j\max} + v_j^l(k+i-1),\ u_{j\min}) \end{aligned}\right\} \text{ if } i=m. \tag{32}$$

The above bounds define the region of values of $v_j^l(k+i)$ which will produce a feasible solution. This definition is followed by the generation of a random binary number b. Based on the value of $b, v_j^l(k+i)$ is modified by the following equations:

$$v_j^l(k+i) = v_j^l(k+i) + (b_u - v_j^l(k+i))(1 - r^{(1-iter)/M}) \quad \text{if } b=0 \tag{33}$$

$$v_j^l(k+i) = v_j^l(k+i) - (v_j^l(k+i) - b_l)(1 - r^{(1-iter)/M}) \quad \text{if } b=1 \tag{34}$$

where r is a random number in [0,1], *iter* is the number of iterations performed so far and M is the expected final number of iterations.

Step 8: If the maximum allowed time has not expired set $iter = iter + 1$ and return the algorithm to step 2. Otherwise stop the algorithm and select the chromosome that produced the lowest value of the objective function throughout the entire procedure.

Remark 5.1. In step 5 it is possible that one chromosome will be selected more than once. This will not cause any problem in the continuation of the algorithm. On the contrary, the chromosome corresponding to the lowest value of the objective function, should be allowed to produce multiple copies in the new generation, which can lead to even better solutions with the application of the genetic operations.

Remark 5.2. The mutation operation described in step 7 is called non-uniform mutation. This operator searches the input space uniformly during the first few iterations, but in later iterations the search becomes more local. The non-uniform mutation is chosen over the uniform one, since the genetic algorithm will locate the region of the global optimum in the first few iterations and in the last ones, it will try to approximate it as close as possible.

Remark 5.3. Theoretical results have been published [28] which show that the genetic algorithms converge to the global optimum, when the best solution is maintained in the population. However, in the convergence analysis the required number of iterations is not specified. Obviously, in the limited amount of time (time between two samples) that is available for running the proposed algorithm, finding the global optimum is not guaranteed. However, due to its probabilistic nature, the algorithm can find a solution, which is approximately optimal. This is perfectly acceptable for the type of problem we try to solve, since the exact global optimum is not necessary.

Contrary to that, the application of traditional optimization techniques to solve the same problem cannot guarantee even the calculation of a feasible solution because of the complexity of the optimization problem, which is due to the following reasons:

- The non-linear fuzzy sets which are used in general
- The fact that in order to code the if-then statements of the fuzzy rules in the formulation of a standard optimization problem, we need to introduce a number of binary decision variables.

The result is the formulation of a complex mixed integer and non-linear programming (MINLP) problem, which is difficult to solve in a reasonable time period, even by today's high computational speed.

Remark 5.4. The genetic algorithm can be accelerated by selecting some chromosomes in step 1 not randomly, but using some information which may be available regarding the distribution of the potential optima. For example, at each time step we can store the solution produced by the genetic algorithm, shift it by one position and give it as an initial chromosome in the next step, where only the input values at the end of the control horizon must be selected randomly. This chromosome might be a very good guess for the solution of the next optimization problem, especially if the system is close to the desired steady-state.

4. A modeling methodology based on the subtractive clustering technique

The identification method which will be used to generate dynamical fuzzy models is based on the subtractive clustering (SC) technique [6] and assumes the availability of K input–output training examples $(\mathbf{x}(i), \mathbf{y}(i))$ $(i = 1, 2, \dots, K)$. The method starts by generating a number of clusters in the $(ni \cdot N)$ dimensional input space. The SC method considers each data point as a potential cluster center and uses a measure of the potential of each data point, which is defined as a function of the

Euclidean distances to all other input data points:

$$P(i)=\sum_{j=1}^{K} \mathrm{e}^{-\frac{4\|\mathbf{x}(i)-\mathbf{x}(j)\|^2}{r_a^2}} \qquad i=1,\ldots,K, \tag{35}$$

where r_a is a radius, defining a neighborhood, which has considerable influence on the potential. Obviously, the potential of a data point is high when it is surrounded by many neighboring data. Given the above definition, the SC algorithm is defined as follows:

Step 1: For $i=1,\ldots,K$ calculate the potential values $P(i)$.
Step 2: Set $L=1$ and select the data point with the highest potential value as the first cluster center. Let $\mathbf{x}^*(1)$ be the location of that point and $P^*(1)$ its potential value.
Step 3: Revise the potential of each data point by the formula:

$$P(i)=P(i)-P^*(1)\mathrm{e}^{-(4\|\mathbf{x}(i)-\mathbf{x}^*(1)\|^2)/r_\beta^2} \quad i=1,\ldots,K, \tag{36}$$

where r_β is also a radius, typically greater than r_a in order to limit the number of generated clusters.
Step 4: Set $L=L+1$ and select the data point with the highest potential value as the next cluster center. Let $\mathbf{x}^*(L)$ be the location of the new cluster and $P^*(L)$ its potential value.
Step 5: Revise the potential of each data point by the formula:

$$P(i)=P(i)-P^*(L)\mathrm{e}^{-(4\|\mathbf{x}(i)-\mathbf{x}^*(L)\|^2)/r_\beta^2} \quad i=1,\ldots,K. \tag{37}$$

Step 6: If the inequality

$$P^*(L) < \varepsilon P^*(1) \tag{38}$$

is true stop the algorithm, else turn the algorithm to step 4. The parameter ε in Eq. (38) is a design parameter, which controls the number of generated clusters.

A complete fuzzy system identification algorithm can be developed based on the results of the SC technique. More specifically a number of L Takagi–Sugeno type fuzzy rules (Eq. 4) can be generated, where the premise parts are fuzzy sets, defined by the cluster centers that are obtained by the SC algorithm. The membership function $A^l(\mathbf{x}(k))$ of an input vector $\mathbf{x}(k)$ to a cluster $\mathbf{x}^*(l)$ can be defined as follows:

$$A^l(\mathbf{x}(k))=\mathrm{e}^{-(4\|\mathbf{x}(k)-\mathbf{x}^*(l)\|^2)/r_a^2}. \tag{39}$$

To this end, the fuzzy system is inferred by the weighted average of the consequent parts of the fuzzy rules ($l=1,2,\ldots,L$) as follows:

$$\hat{\mathbf{y}}(k)=\frac{\sum_{l=1}^{L}[A^l(\mathbf{x}(k))(\mathbf{h}^l+\mathbf{H}_1^l\mathbf{u}(k-1)+\mathbf{H}_2^l\mathbf{u}(k-2)+\cdots+\mathbf{H}_N^l\mathbf{u}(k-N))]}{\sum_{l=1}^{L}A^l(\mathbf{x}(k))}. \tag{40}$$

In the next step we define the lth Fuzzy Basis Function (FBF) as,

$$p^l(\mathbf{x}(k))=\frac{A^l(\mathbf{x}(k))}{\sum_{l=1}^{L}A^l(\mathbf{x}(k))} \tag{41}$$

and Eq. (40) is transformed into the following FBFs expansion,

$$\begin{aligned}\hat{\mathbf{y}}(k) &= \sum_{l=1}^{L} [p^l(\mathbf{x}(k))(\mathbf{h}^l + \mathbf{H}_1^l\mathbf{u}(k-1) + \mathbf{H}_2^l\mathbf{u}(k-2) + \cdots + \mathbf{H}_N^l\mathbf{u}(k-N))] \\ &= \sum_{l=1}^{L} [p^l(\mathbf{x}(k))(\mathbf{h}^l + \mathbf{G}^l\mathbf{x}(k))],\end{aligned} \tag{42}$$

where

$$\mathbf{G}^l = [\mathbf{H}_1^l; \mathbf{H}_2^l; \ldots; \mathbf{H}_N^l]. \tag{43}$$

As shown in Eq. (42), the FBF expansion of the fuzzy system gives a linear expression with respect to the unknown parameters and enables the utilization of classical linear optimization techniques to determine their optimal values. More specifically from Eq. (42) and for $i = 1, \ldots, no$ we can write:

$$\begin{bmatrix} \hat{y}_i(1) \\ \vdots \\ \hat{y}_i(K) \end{bmatrix} = \begin{bmatrix} p^1(\mathbf{x}(1)) & p^1(\mathbf{x}(1))\mathbf{x}^{\mathrm{T}}(1) & \cdots & p^L(\mathbf{x}(1)) & p^L(\mathbf{x}(1))\mathbf{x}^{\mathrm{T}}(1) \\ & & \vdots & & \\ p^1(\mathbf{x}(K)) & p^1(\mathbf{x}(K))\mathbf{x}^{\mathrm{T}}(K) & \cdots & p^L(\mathbf{x}(K)) & p^L(\mathbf{x}(K))\mathbf{x}^{\mathrm{T}}(K) \end{bmatrix} \begin{bmatrix} h_1(i) \\ \mathbf{G}_1(i,:)^{\mathrm{T}} \\ \\ h_L(i) \\ \mathbf{G}_L(i,:)^{\mathrm{T}} \end{bmatrix} \tag{44}$$

A different equation can be written for each output variable. Replacing the vector in the left-hand side of the above equation with the actual output values of the K training examples, the problem reduces to a simple least square problem, which obtains the unknown parameters of the consequent parts of the fuzzy rules.

5. Case study

The proposed methodology was used to model and control a non-isothermal continuous stirred tank reactor (CSTR), which is characterized by the following dynamic equations:

$$\begin{aligned}\frac{\mathrm{d}C_{\mathrm{A}}}{\mathrm{d}t} &= \frac{F}{V}(C_{\mathrm{A,in}} - C_{\mathrm{A}}) - 2k_o \exp\left(-\frac{E}{RT}\right) C_{\mathrm{A}}^2, \\ \frac{\mathrm{d}T}{\mathrm{d}t} &= \frac{F}{V}(T_{\mathrm{in}} - T) + 2\,\frac{(-\Delta H)_{\mathrm{R}}}{\rho c_{\mathrm{p}}}\,k_o \exp\left(-\frac{E}{RT}\right) C_{\mathrm{A}}^2 - \frac{UA}{V\rho c_{\mathrm{p}}}(T - T_j),\end{aligned} \tag{45}$$

where V is the volume of the CSTR; $(\Delta H)_{\mathrm{R}}$ is the heat of the reaction; and $-E/R$, k_o, c_{p}, ρ are constants of the reaction and the reactants. The variables $F, C_{\mathrm{A,in}}$, T_{in} and T_j are considered as inputs to the system, while C_{A}, and T are the outputs. F is the flow rate into the reactor, $C_{\mathrm{A,in}}$ is the inlet concentration of A, C_{A} is the concentration of A inside the reactor, T_{in} is the inlet temperature, T is the temperature inside the reactor and T_j is the temperature of the coolant. The values of the process parameters are shown in Table 1.

The described modeling methodology was applied to the SISO system defined by Eq. (45) where all the input variables besides the coolant temperature are considered as disturbances and the

Table 1
Process parameters in the CSTR example

Process parameters	Value
V	100 l
UA	20000 J/ min K
ρ	1000 g/l
C_p	4.2 J/g K
$-(\Delta H)_\mathrm{R}$	596619 J/mol
k_o	6.85E + 11 l/min mol
E	76534.704 J/mol
R	8.314 J/mol K

Table 2
Steady state values of the input variables in the CSTR example

Input variable	Steady state
F	20 l/min
T_in	275 K
T_j	250 K
$C_\mathrm{A,\,in}$	1 mol/l

temperature inside the reactor is considered as the only controlled variable. The method can be easily extended to a MIMO system where all the input variables can be manipulated to control both output parameters.

In order to develop a model for the SISO system we created a set of 1000 data examples by adding random number signals to the steady state values of the cooling temperature, keeping the values of all the other input variables at the steady states shown in Table 2. The sampling period for these signals was equal to 1 min. The input and output data were scaled between -1 and 1 and partitioned into two subsets. The first 500 points were used for training the model and the rest of the data were used for validation. The input vector to the dynamic fuzzy model consisted of 10 past values of the cooling temperature. This selection resulted in the formulation of a Non-linear Finite Impulse Response (NFIR) type of model where no past values of the output variables are used in the input vector [1,16]. The inclusion of previous values of the output variable would result in the formulation of a non-linear auto regressive with eXogenous inputs (NARX) model, which is more compact, but is more possible to fail, when it is applied to data collected from a real chemical process. The reason is that true data are usually rather noisy and in this case the identification algorithms produce poor NARX models, since they give much more importance to the terms associated with the previous values of the output variables compared to the terms associated with the input variables.

The subtractive clustering method was used to develop a dynamic fuzzy model, with $r_a = 3$, $r_\beta = 4.5$ and $\varepsilon = 0.1$. The produced dynamic model consisted of 17 rules. The model predictions along with the actual values for 200 data points of the second data set can be found in Fig. 3, proving the validity of the method. The sum of squared errors (SSE) corresponding to the full validation data set was 939.96.

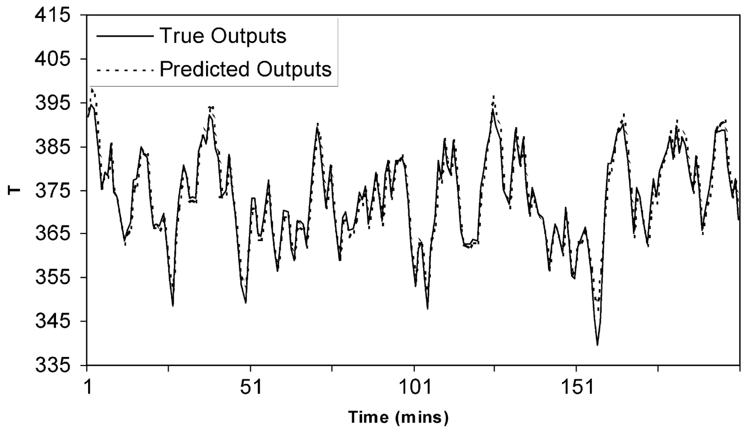


Fig. 3. Actual values and fuzzy model predictions for temperature inside the CSTR.

The dynamic fuzzy model was used as the basis for applying the proposed FMPC methodology, using the norm-1 formulation of the objective function (Eq. 8). The control horizon and the prediction horizon were both set to predetermined values based on some general tuning rules, which have been published in the literature. Using these rules, the prediction horizon was selected to be equal to the settling time of the process. The control horizon was set equal to half of the prediction horizon for good robust performance. The values of the output weights and the move suppression parameters were selected by trial and error taking into account that utilization of large values for the move suppression coefficients, increases the robust stability of the closed loop system but deteriorates the performance. Based on these rules, the FMPC parameters were selected as follows:

Prediction horizon: $p=10$.
Control horizon: $m=5$.
Output weights: $\Theta=1$.
Move suppression coefficients: $R_i=2$.

The parameters of the genetic algorithm should be selected so that there is a considerable diversification in the population during each iteration, but at the same time a random search is avoided. With a proper selection of the parameter values, the successful chromosomes are maintained through the generations and the speed of convergence is accelerated. The De Jong rules [15] were followed which suggest the choice of a high crossover probability, a low mutation probability (inversely proportional to the population size) and a moderate population size. Following these rules, 20 chromosomes were used to formulate the population of the genetic algorithm. The probabilities of crossover p_c and mutation p_m were set equal to 0.3 and 0.05, respectively. As far as the number of iterations is concerned, during each time interval the genetic algorithm was allowed to run for 1 min. This is equal to the time that is available for calculating the next control action and corresponds to about 300 iterations.

The simulation presented next introduces a step disturbance to the feed to the reactor from 20 to 15 l/min and the controller needs to return the temperature inside the reactor to the desired set point

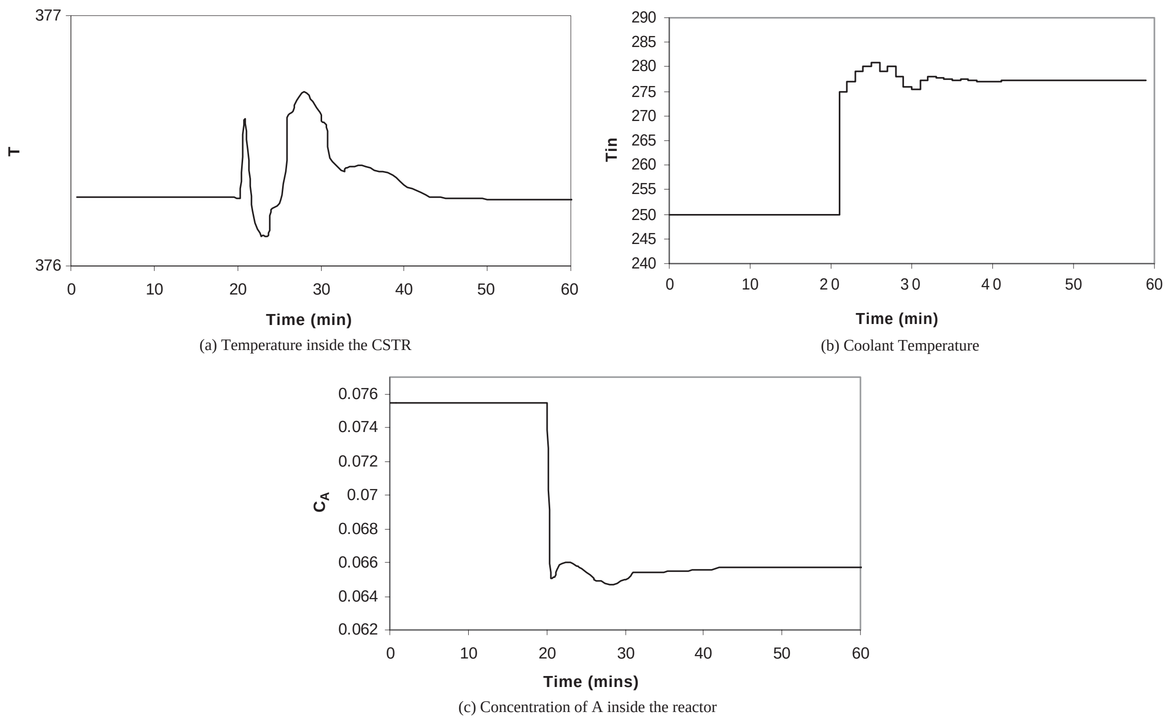
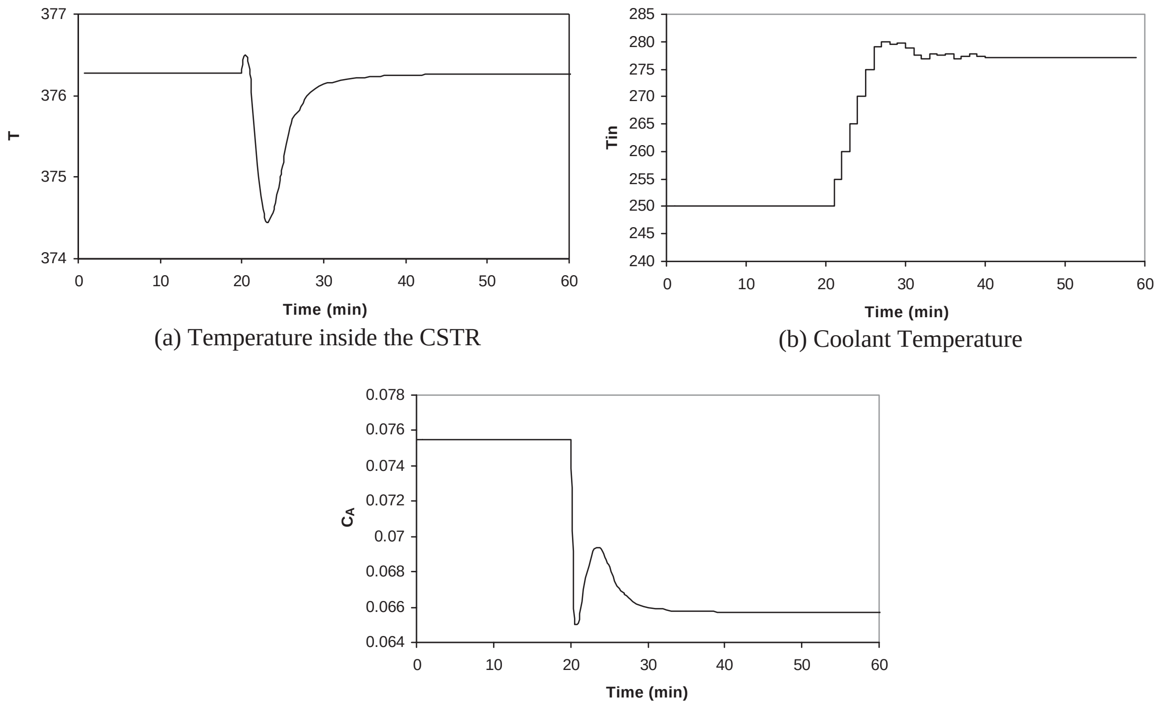


Fig. 4. FMPC input and output responses when no input move constraints are posed.

(376.3 K). Two cases were examined: the case where no constraints are posed on the input moves of the manipulated variables and the case where the temperature difference of the coolant between two subsequent time points cannot exceed 5° K. The controller performed successfully in both cases and the temperature inside the reactor returned to the desired set point in a very short time. Obviously the controller was slower and more conservative in the second case. However, the controller used correctly the maximum possible move (−5 K) to drive the input variable to the right direction in the minimum possible time.

The input and output behavior for both cases can be found in Figs. 4 and 5, respectively. For the completion of the study, the responses of the concentration C_A are shown as well. It is also very interesting to watch the lowest value of the objective function achieved by the genetic algorithm as a function of time. This is given in Fig. 6 for the 21st and 32nd time points, where the best value of the objective function is plotted against the number of iterations. Both graphs refer to the simulation, where constraints were posed on the moves of the input variable. We observe that when the system is close to the desired steady state, the genetic algorithm almost reached the optimum value of the objective function only in a few iterations. It is interesting to notice that although the algorithm performed about 300 iterations, the same results could be achieved if only 100 iterations were allowed for each control move calculation.

The proposed technique was compared to the standard LDMC methodology, which consists of Eqs. (8) and (10)–(13), with the only difference that the future behavior of the process is expressed



(a) Temperature inside the CSTR (b) Coolant Temperature

(c) Concentration of A inside the reactor

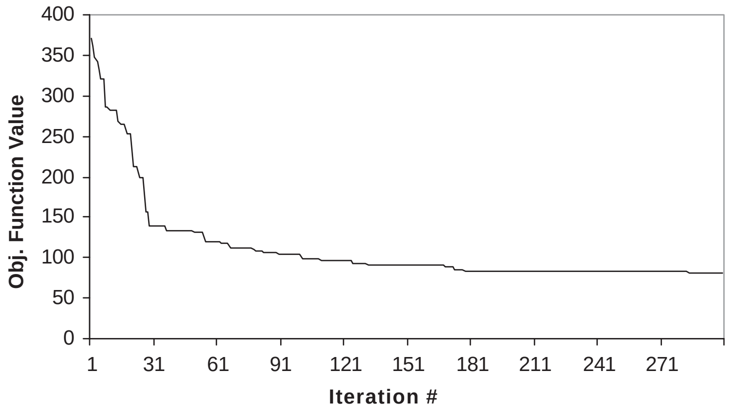
Fig. 5. FMPC input and output responses when input move constraints are posed.

by a linear finite impulse response (FIR) model instead of a non-linear fuzzy model. More specifically Eq. (7) in the case of a linear FIR model can be written as follows:

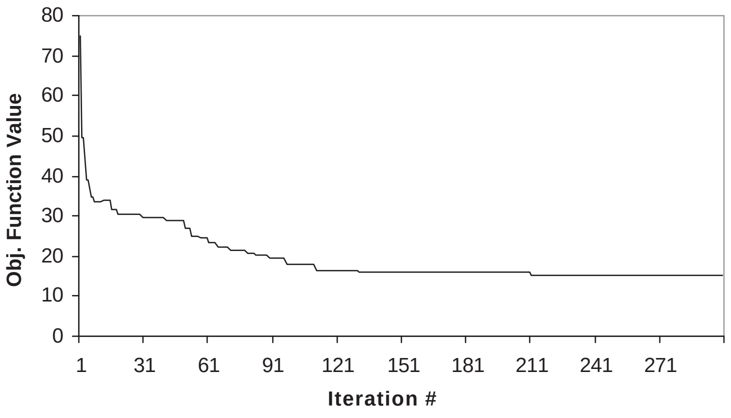
$$\hat{\mathbf{y}}(k\,|\,k) = \mathbf{S}_1\mathbf{u}(k-1) + \mathbf{S}_2\mathbf{u}(k-2) + \cdots + \mathbf{S}_N\mathbf{u}(k-N), \tag{46}$$

where $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_N$ are matrices with dimensions $(no \times ni)$. For the particular example with one input variable and one output variable, each one of those matrices consists of only one element. Again, 10 past values of the coolant temperature were used to formulate the input vector of the model. The values of the elements were determined by using linear regression on the same data set that was utilized for the development of the Takagi–Sugeno model. A comparison of the predicting abilities of the linear model against the fuzzy model can be easily achieved by applying the linear FIR model on the validation data. The SSE corresponding to the full validation data set was found to be 3084.1 and the model predictions along with the actual data for 200 testing points are shown in Fig. 7. These results proved the superiority of the fuzzy modeling technique.

The linear model was used as a predictor in the LDMC scheme. The two scenarios used for testing the proposed FMPC method were also simulated using the LDMC scheme, and produced the trajectories shown in Figs. 8 and 9. The performance of the FMPC method is superior to the one of the LDMC method in both cases, since it results in responses, which are closer to the desired



(a) k=21



(b) k=32

Fig. 6. The role of iterations in finding the best possible objective function value.

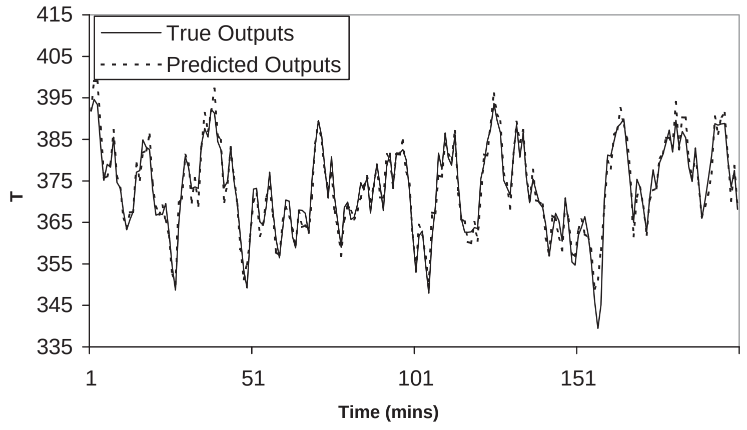


Fig. 7. Actual values and linear model predictions for temperature inside the CSTR.

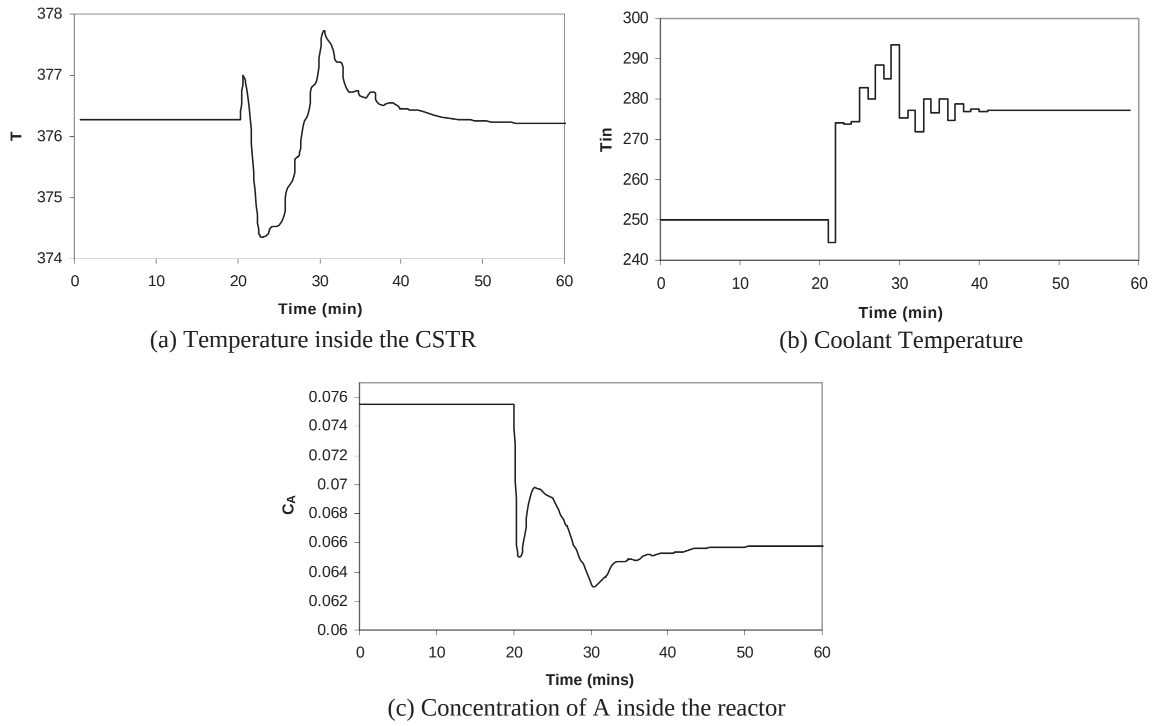


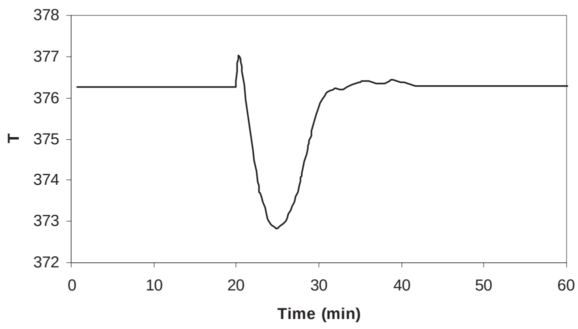
Fig. 8. LDMC input and output responses when no input move constraints are posed.

steady state. The advantage of the LDMC method is that it formulates a trivial LP problem, where the calculation of the global optimum in a very short period of time is guaranteed.

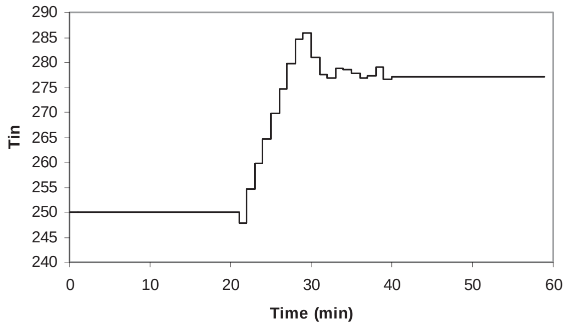
6. Conclusions

A generic FMPC structure was proposed, where any fuzzy model can serve as the predictor of the future behavior of the process. The method can be used to control highly non-linear systems and since it utilizes fuzzy systems to simulate the dynamics of the process, it can use both information hidden in process data and knowledge provided by human experts to improve the prediction capabilities of the model and the performance of the control strategy. The proposed technique formulates an optimization problem that needs to be solved in real time. For the solution of this rather complicated non-linear optimization problem, a genetic algorithm is proposed, which can approximate the optimum solution very fast, compared to conventional optimization techniques. The genetic algorithm proves to be very efficient in the solution of the optimization problem since compliance with the input and input move constraints is guaranteed by the proper selection of chromosomes in the initial population and the carefully designed genetic operators.

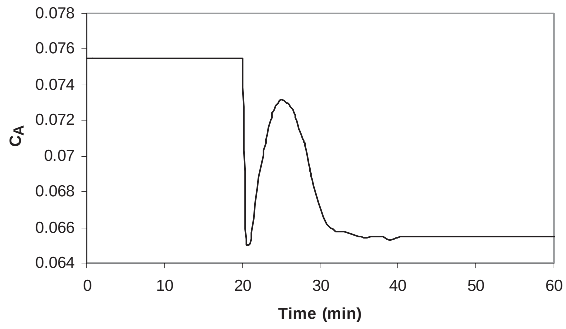
The method was applied to a chemical reactor with one input variable and one output variable, using a discrete Takagi–Sugeno fuzzy model for predicting the behavior of the process. The methodology produced very good results, even when we posed constraints on the control moves and proved to be superior to the standard LDMC approach. Future research is going to investigate if there is



(a) Temperature inside the CSTR



(b) Coolant Temperature



(c) Concentration of A inside the reactor

Fig. 9. LDMC input and output responses when input move constraints are posed.

any rigorous way to select the controller parameters, so that the performance of the controller is improved without leading to stability problems.

References

[1] J. Abonyi, A. Bodizs, L. Nagy, F. Szeifert, Hybrid fuzzy convolution model and its application in model predictive control, Chem. Eng. Res. Des. 78 (A) (2000) 597–604.
[2] R.A. Abou-Jeyab, Y.P. Gupta, J.R. Gervais, P.A. Branchi, S.S. Woo, Constrained multivariable control of a distillation column using a simplified model predictive control algorithm, J. Process Control 11 (2001) 509–517.
[3] R.M. Ansari, M.O. Tade, Non-linear Model Based Process Control: Applications in Petroleum Refining, Springer, London, England, 2000.
[4] J.G. Bekker, I.K. Craig, P.C. Pistorius, Model predictive control of an electric arc furnace off-gas process, Control Eng. Practice 8 (2000) 445–455.
[5] W.V. Brempt, T. Backx, J. Ludlage, P.V. Overschee, B.D. Moor, R. Tousain, A high performance model predictive controller: application on a polyethylene gas phase reactor, Control Eng. Practice 9 (2001) 829–835.
[6] S.L. Chiu, Fuzzy model identification based on cluster estimation, J. Intell. Fuzzy Systems 2 (1994) 267–278.
[7] D.W. Clarke, C. Mohtadi, P.S. Tuffts, Generalized predictive control: part I, The basic algorithm, Automatica 23 (2) (1987) 137–148.
[8] D.W. Clarke, C. Mohtadi, P.S. Tuffts, Generalized predictive control: part I, The basic algorithm, Automatica 23 (2) (1987) 149–160.
[9] C.R. Cutler, B.L. Ramaker, Dynamic matrix control—A computer control algorithm, Joint Automatic Control Conf., San Francisco, CA, 1980.

[10] M. Fisher, O. Nelles, R. Isermann, Adaptive predictive control of a heat exchanger based on a fuzzy model, Control Eng. Practice 6 (1998) 259–269.
[11] P.J. Fleming, R.C. Purshouse, Evolutionary algorithms in control system engineering: a survey, Control Eng. Practice, in press.
[12] C.E. Garcia, M. Morari, Internal model control. 1, A unifying review and some new results, Ind. Eng. Chem. Process. Des. Dev. 21 (2) (1982) 308–323.
[13] C.E. Garcia, A.M. Morshedi, Quadratic programming solution of dynamic matrix control (QDMC), Chem. Eng. Comm. 46 (1986) 73–87.
[14] A. Geyer-Schulz, Fuzzy Rule-Based Expert Systems and Genetic Machine Learning, Physica-Verlag, Heidelberg, Germany, 1996.
[15] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[16] Y.L. Huang, H.H. Lou, J.P. Gong, T.F. Edgar, Fuzzy model predictive control, IEEE Trans. Fuzzy Systems 8 (2000) 665–677.
[17] C.L. Karr, E.J. Gentry, Fuzzy control of pH using genetic algorithms, IEEE Trans. Fuzzy Systems 1 (1) (1993) 46–53.
[18] S.W. Kim, E.T. Kim, M. Park, A new adaptive fuzzy controller using the parallel structure of fuzzy controller and its application, Fuzzy Sets and Systems 81 (1996) 205–226.
[19] J. Kim, Y. Moon, B.P. Zeigler, Designing fuzzy net controllers using genetic algorithms, IEEE Control System Mag. 15 (1995) 62–72.
[20] R.K. Mehra, R. Rouhani, Theoretical considerations on model algorithmic control for nonminimum phase systems, Joint Automatic Control Conf., San Francisco, CA, 1980.
[21] Z. Michalewitz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, Berlin, Germany, 1992.
[22] H. Moriyama, K. Shimizu, On-line optimization of culture temperature for ethanol fermentation using a genetic algorithm, J. Chem. Technol. Biotechnol. 66 (1996) 217–222.
[23] M. Morshedi, C.R. Cutler, T.A. Skrovanek, Optimal solution of dynamic matrix control with linear programming techniques (LDMC), Proc. Amer. Control Conf., Boston, MA, 1985.
[24] J.V. de Oliveira, J.M. Lemos, Long-range adaptive fuzzy relational control, Fuzzy Sets and Systems 70 (1995) 337–357.
[25] C. Onnen, R. Babuška, U. Kaymak, J.M. Sousa, H.B. Verbruggen, R. Isermann, Genetic algorithms for optimization in predictive control, Control Eng. Practice 5 (1997) 1363–1372.
[26] D.M. Prett, R.D. Gilette, Optimization and constrained multivariable control of a catalytic cracking unit, AIChE Annual Meeting, Houston, TX, 1979.
[27] J. Richalet, A. Rault, J.L. Testud, J. Papon, Model predictive heuristic control, Application to industrial processes, Automatica 14 (1978) 413–428.
[28] G. Rudolph, Convergence analysis of canonical genetic algorithms, IEEE Trans. Neural Networks 5 (1) (1994) 96–101.
[29] E. Sanchez, T. Shibata, L.A. Zadeh (Eds.), Genetic Algorithms and Fuzzy Logic Systems: Soft Computing Perspectives, World Scientific, River Edge, NJ, 1997.
[30] K. Shimojima, T. Fukuda, Y. Hasegawa, Self-tuning modeling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm, Fuzzy Sets and Systems 71 (1995) 295–309.
[31] J.M. Sousa, R. Babuska, H.B. Verbuggen, Fuzzy predictive control applied to an air-conditioning system, Control Eng. Practice 5 (1997) 1395–1406.
[32] A. Tsez, P.Y. Peng, J. Guthy, Genetic-based fuzzy clustering for DC-motor friction identification and compensation, IEEE Trans. Control Systems Technol. 6 (1998) 462–472.
[33] A. Varšek, T. Urbančič, B. Fillipič, Genetic algorithms in controller design and tuning, IEEE Trans. Systems Man Cybernet. 23 (1993) 1330–1339.
[34] P.D. Vucovic, One step ahead predictive fuzzy controller, Fuzzy Sets and Systems 122 (2001) 107–115.
[35] L.X. Wang, Adaptive Fuzzy Systems and Control, Prentice-Hall, Englewood Cliffs, NJ, 1994.