Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2015, Article ID 437943, 15 pages
http://dx.doi.org/10.1155/2015/437943

Research Article

Constrained Fuzzy Predictive Control Using Particle Swarm Optimization

Oussama Ait Sahed,[1] Kamel Kara,[1] and Mohamed Laid Hadjili[2]

[1]*SET Laboratory, Electronics Department, University of Blida 1, Route de Soumaa, BP 270, 09000 Blida, Algeria*
[2]*High School of Computer Sciences (HEB-ESI), Rue Royale 67, 1000 Brussels, Belgium*

Correspondence should be addressed to Oussama Ait Sahed; aitsahed.oussama@hotmail.com

Received 26 September 2014; Revised 24 April 2015; Accepted 24 April 2015

Academic Editor: Shyi-Ming Chen

A fuzzy predictive controller using particle swarm optimization (PSO) approach is proposed. The aim is to develop an efficient algorithm that is able to handle the relatively complex optimization problem with minimal computational time. This can be achieved using reduced population size and small number of iterations. In this algorithm, instead of using the uniform distribution as in the conventional PSO algorithm, the initial particles positions are distributed according to the normal distribution law, within the area around the best position. The radius limiting this area is adaptively changed according to the tracking error values. Moreover, the choice of the initial best position is based on prior knowledge about the search space landscape and the fact that in most practical applications the dynamic optimization problem changes are gradual. The efficiency of the proposed control algorithm is evaluated by considering the control of the model of a 4×4 Multi-Input Multi-Output industrial boiler. This model is characterized by being nonlinear with high interactions between its inputs and outputs, having a nonminimum phase behaviour, and containing instabilities and time delays. The obtained results are compared to those of the control algorithms based on the conventional PSO and the linear approach.

1. Introduction

In the late seventies, a new class of advanced control algorithms, grouped under the denomination Model Predictive Control (MPC), has emerged. The MPC strategy is based on the use of an explicit model to predict the process future behaviour over a finite horizon and then to compute a sequence of future control actions by minimizing a given cost function. The MPC is known to be a very powerful control strategy for many industrial processes [1–3]. Its attraction is due to its ability to handle complex control problems which involve multivariable process interactions, constraints in the system variables, nonminimum phase behaviour, and variable or unknown time delays. Linear MPC (LMPC) techniques, which use a linear model for the process, have been successfully used in many industrial applications, such as refining, chemicals, petrochemicals, air and gas processing, and food processing industries [4–6].

Although LMPC algorithms provide satisfactory performance in many applications, against highly nonlinear process, severe degradation in the control performance can occur, unless of course the operating conditions are very close to the steady state around which the model is linearized [7]. Since most practical processes are nonlinear by nature, new efficient nonlinear MPC (NMPC) techniques have to be derived to incorporate nonlinearities and ensure higher control performance. The introduction of nonlinear models will enhance the overall controller performance. However, the relatively simple optimization problem in LMPC algorithms will be transformed to a nonlinear and a nonconvex optimization problem which requires complex and time consuming procedures [8–10]. Thus, the challenge is to derive an efficient optimization algorithm that allows the real-time implementation of the control law.

An important step in the design of NMPC algorithms based controllers is the construction of the explicit nonlinear

model of the system. The model choice is a critical task; the controller efficiency and performance depend extremely on the accuracy of the used model. In many cases it is even impossible to obtain a suitable model for the underlying process due to its complexity or lack of knowledge of its critical parameters. Different approaches that can provide satisfactory models exist; one which is widely used is the fuzzy logic technique [11]. Fuzzy inference systems (FIS) are universal approximators capable of approximating any continuous function with a certain level of accuracy [12]. Takagi-Sugeno (TS) models, a subdivision of fuzzy models, are particularly suitable for NMPC algorithms [13]. These models are able to express the dynamic nature of systems with characteristics of randomness, large delay time, and strong nonlinearity [14, 15].

Another important factor in driving the NMPC based controllers is the resolution of the relatively difficult and time consuming nonlinear and nonconvex optimization problem. Seeing that the analytical approach is impractical, several other suboptimal solutions were proposed [7, 16–18]. Some of them are based on numerical optimization algorithms, which are known to have a slow convergence rate and could easily be trapped in local minima. Other more interesting approaches are the metaheuristic based optimization algorithms, such as the powerful but computationally complex genetic algorithms (GA) and the particle swarm optimization strategy. PSO algorithms require much less computational effort than the GA and their implementation is relatively easy [19]. A lot of papers have considered the PSO to solve the optimization problem of NMPC [20–23]. In these papers, a large number of particles and iterations were used which increases the computational burden of the control algorithm. Recently, several PSO algorithms have been proposed in the literature to solve different optimization problems [24–26]. In most of these algorithms, improving the accuracy and the convergence proprieties of the algorithm is obtained at the cost of increasing its computing complexity. For example, in [26] more than one equation is used to update the particles positions at the same time. Obviously, using several update equations will increase the algorithm computing time and will limit its use in real time applications.

In this work, a PSO based constrained fuzzy predictive controller is proposed. The aim is to develop a simple and efficient controller capable of handling the relatively complex optimization problem with minimal computational time. This can be achieved using a reduced population size and a small number of iterations. The proposed control algorithm uses the normal distribution law to distribute the initial particles positions within the area around the best position. The radius limiting this area is adaptively changed according to the tracking error values. Moreover, the choice of the initial best position is based on prior knowledge about the search space landscape and the fact that in most practical applications the dynamic optimization problem changes are gradual. The algorithm efficiency is evaluated by considering the control of the model of a 4 × 4 Multi-Input Multi-Output (MIMO) industrial boiler [27, 28]. This model is characterized by being nonlinear with high interactions between its inputs and outputs, having a nonminimum phase behaviour, containing instabilities and time delays, and including constraints on its variables, disturbances, and uncertainties.

The paper is organized as follows. In Section 2 the design stages of Takagi-Sugeno fuzzy models are presented. Section 3 presents and describes the fuzzy predictive control problem. The PSO algorithm, used to solve the fuzzy MPC optimization problem, is introduced in Section 4. Section 5 deals with the proposed control algorithm and Section 6 gives the simulation results. Finally, conclusions are drawn in the last section.

2. Takagi-Sugeno Fuzzy Modelling

Fuzzy inference systems (FIS) are universal approximators capable of approximating any continuous function with certain level of accuracy [12]. The FIS are based on the so-called IF-THEN rules and are classified into two main categories: Mamdani models and Takagi-Sugeno models. The rules of TS fuzzy models, used in this work, have the following form:

$$R_j\text{: IF } \mathcal{F} \text{ is } A_j \text{ THEN } y_j = \theta_j^T \mathcal{X}, \tag{1}$$

where $\mathcal{F} = \{y(t), y(t-1)\cdots y(t-n_a+1), u(t), u(t-1)\cdots u(t-m_a+1)\}$ is the set of $n_a + m_a$ premise values, $\mathcal{X}^T = \{y(t), y(t-1)\cdots y(t-n_r+1), u(t), u(t-1)\cdots u(t-m_r+1), 1\}$ is the set of m_r inputs and n_r outputs values used in the consequent regressors, $A_j = \{A_{1,j}, A_{2,j}\cdots A_{n_a,j}, B_{1,j}, B_{2,j}\cdots B_{m_a,j}\}$ is the set of membership functions associated with the antecedents of the jth rule, and $\theta_j^T = \{a_{j,1}, a_{j,2}\cdots a_{j,n_r}, b_{j,1}, b_{j,2}\cdots b_{j,m_r}, c_j\}$ is the parameter vector of the jth submodel while y_j is its output ($1 \le j \le r$; r is the number of fuzzy rules).

It is clear, from (1), that the Takagi-Sugeno fuzzy models have fuzzy propositions only in their antecedents while their consequences are linear functions of the antecedents or their variables.

Using input/output data extracted from a nonlinear process, a fuzzy model can be constructed to mimic the process behaviour, by defining the parameter matrix as

$$\Theta = \left[\theta_1, \theta_2 \cdots \theta_r\right] \tag{2}$$

and the normalized membership grade vector as

$$\mu^T = \left[\mu_1, \mu_2 \cdots \mu_r\right]. \tag{3}$$

The inferred output of the TS fuzzy model is given by

$$\hat{y}(t+1) = \left(\Theta\mu\right)^T \chi. \tag{4}$$

Using the input/output representation, the TS model given by (4) can be rewritten as follows:

$$\begin{aligned}\hat{y}(t+1) = &\sum_{p=1}^{n_a} \overline{a}_p(t)\, y(t-p+1) \\ &+ \sum_{p=1}^{m_a} \overline{b}_p(t)\, u(t-p+1) + \overline{c}(t),\end{aligned} \tag{5}$$

where $\overline{a}_p(t) = \sum_{l=1}^{r} \mu_l(t) a_{l,p}$, $\overline{b}_p(t) = \sum_{l=1}^{r} \mu_l(t) b_{l,p}$, and $\overline{c}(t) = \sum_{l=1}^{r} \mu_l(t) c_l$.

The global output of the fuzzy model can be written as

$$\begin{aligned}
&\widehat{y}(t+1) \\
&= \sum_{p=1}^{r} \mu_p(t)\, y_p(t+1) \\
&= \left[\mu_1(t)\, y(t), \ldots, \mu_1(t)\, u(t-m_r+1), \ldots, \mu_r(t)\right] \widetilde{\theta} \\
&= \varphi^T(t)\, \widetilde{\theta},
\end{aligned} \tag{6}$$

where $y_p(t+1)$ is the output of the pth submodel and

$$\widetilde{\theta}^T = \left[\theta_1^T \;\; \theta_2^T \;\; \cdots \;\; \theta_r^T\right]. \tag{7}$$

Assuming that a set of N input-output data pairs $(u(t), y(t))$ is available and defining $L = \max\{m_r, n_r\}$, a regression matrix which has the following expression can be constructed:

$$\Phi = \begin{bmatrix} \varphi^T(L) \\ \varphi^T(L+1) \\ \vdots \\ \varphi^T(N-1) \end{bmatrix}. \tag{8}$$

The vector $\widetilde{\theta}$ can be calculated by solving the following least square problem:

$$Y = \Phi \widetilde{\theta}, \tag{9}$$

where $Y = [y(L+1), \ldots, y(N)]^T$.

The resolution of (9), which is the last step in TS fuzzy model construction, deals with the definition of the consequences part of the rules. However, the antecedent, which involves in part the number, the position, the shape, and the distribution of the membership functions, must be first selected. A lot of techniques for the construction of fuzzy models from input-output data are described in the literature; in Espinosa et al. [16] different approaches are well explained.

3. Fuzzy Predictive Control Principle

We can always distinguish two common features present in any given MPC based algorithm. The explicit model used to predict the future process behaviour and the optimization problem from which a control sequence is derived.

The main steps of the MPC strategy are given below:

(1) The constructed model is used to predict the future behaviour of the process over a given prediction horizon.

(2) The reference trajectory must be defined over the prediction horizon.

(3) The control sequence is obtained by minimizing a given cost function.

(4) Only the first element of the control sequence is applied on the system.

The previous steps are repeated, at every sampling time, according to the receding horizon idea. These steps are illustrated by the block diagram of Figure 1.

The optimization problem to be solved at every sampling time usually has the following form:

$$\begin{aligned}
\min_{\Delta u(t)} \quad & J\left(u(t), \widehat{y}(t), w(t)\right) \\
&= \sum_{j=N_1}^{N_2} \left[\left(\widehat{y}(t+j \mid t) - w(t+j)\right)^T Q \left(\widehat{y}(t+j \mid t) - w(t+j)\right)\right] + \sum_{j=1}^{N_u} \left[\Delta u^T(t+j-1)\, R \Delta u(t+j-1)\right] \\
\text{subject to:} \quad & \Delta u(t+j-1) = 0, \quad \text{for } j > N_u \\
& y_{\min} < \widehat{y}(t+j \mid t) < y_{\max} \quad \text{for } N_1 \le j \le N_2 \\
& u_{\min} < u(t+j) < u_{\max} \quad \text{for } 1 \le j \le N_u \\
& \Delta u_{\min} < \Delta u(t+j-1) < \Delta u_{\max} \quad \text{for } j \le N_u,
\end{aligned} \tag{10}$$

where $\Delta u(t) = u(t) - u(t-1)$ is the control increment, J is the cost function, N_1 and N_2 are, respectively, the minimum and maximum prediction horizons, N_u is the control horizon, Q is a positive definite matrix and R is a positive semidefinite matrix which are called weight matrices, $w(t)$ is the reference trajectory, $u(t)$ represents the system inputs, and $\widehat{y}(t+j \mid t)$ is a j-step ahead prediction of the system outputs on data up to time t. The predicted outputs can recursively be generated using the process fuzzy model or obtained using a multistep prediction model of the process.

Given that the predicted outputs are nonlinear with respect to the control inputs, the optimization problem given by (10) is then a constrained nonlinear and nonconvex optimization problem, the solution of which is difficult and

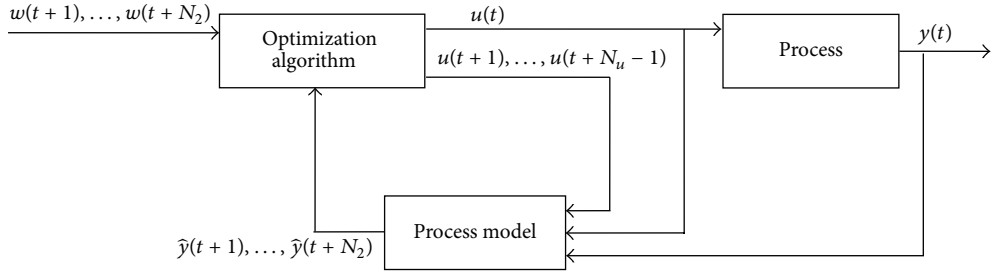

Figure 1: Basic structure of MPC technique.

generally expensive in computing time. Different approaches were investigated to solve this problem, such as the numerical optimization techniques [29–31], the metaheuristic based optimization algorithms [7, 15, 22, 32], the linearization of the process fuzzy model [18], and the use of particular model structures to obtain a convex form for the cost function [33].

4. PSO Algorithm

The particle swarm optimization is a modern and versatile population based optimization technique, in many respects, similar to evolutionary algorithms [19]. It was first introduced by Kennedy and Eberhart in 1995, as a solution for the optimization problem of a single objective continuous problem. It is based on the observations made upon the collective behaviour of social living entities, such as ants, bees, and flocks of birds. Basically, each particle will be moving around the search space looking for an adequate solution to the optimization problem. By evaluating the fitness of each one of them, their suitability as a possible solution will be assessed. In the next iteration, each particle will change its position to provide more accurate solution, by considering the history of the particle or its best position, its neighborhood history, and mainly the best position found by the whole swarm.

The particles movements are bound by the following equations:

$$\begin{aligned} v(m+1) &= v(m) + a(m+1), \\ x(m+1) &= x(m) + v(m+1), \end{aligned} \tag{11}$$

where v, a, x are, respectively, the particle, velocity, acceleration, and position and m is the iteration index.

In the Clerc-Kennedy PSO [34], considered as the conventional PSO algorithm, the particle acceleration has the following expression:

$$a = \chi \left[c\epsilon_1 \left(P_g - x \right) + c\epsilon_2 \left(P - x \right) \right] - \left(1 - \chi \right) v, \tag{12}$$

where $\chi = 0.729843788$, $c = 2.05$, ϵ_1 and ϵ_2 are random numbers from the uniform distribution $U[0,1]$, P is the particle personal best position, and P_g is the global best position of all particles. Parameter χ guarantees a decreasing velocity for each particle as the number of iterations increases. This property will improve the convergence quality as the particles approach the solution; their movement will be more and more limited, which help to fine-tune the found solution.

The PSO population size is extremely dependent on the optimization problem at hand; a compromise must be reached between the size of the population and the global performance. Empirically, it has been shown that a population of about 20 to 30 particles is generally sufficient in handling almost all the classical optimization problems [35]. Some other optimization problems could be efficiently solved by a smaller population or rather require a larger one. Several papers have dealt with this problem. The author in [35] describes an adaptive PSO algorithm where the swarm population size is obtained through different strategies that regulate the population by removing badly performing particles and creating new ones. In [36] the VarPSO has been proposed, where the PSO population size is varied during the run, based on the concepts of particle age and neighborhood.

One of the major drawbacks of the original PSO algorithm is its inability to consistently improve a solution as the number of iterations increases [37]. Several improvements were proposed to address this weakness, such as the tent-map PSO algorithm introduced by Song et al. [38] and the self-organizing hierarchical particle swarm optimization [39]. In [34], a constriction factor χ ($\chi < 1$) is used to progressively contract the particle movement, as the iterations increase. Different distribution laws like exponential and Gaussian have already been used for the fine-tuning of PSO parameters [40, 41]. But for initializing the particles most of the proposed algorithms use the uniform distribution. The authors in [42] have investigated the possibility of using different distribution law to initialize the particles. They have shown that using other distribution laws gives better performances for most of the cases. In the remaining ones, the performances were the same. This is due to the nature of the uniform distribution, which sometimes gives a wrong impression of the relative performance of algorithms as it is shown in [43].

One of the reasons for the poor performance of the basic version of PSO algorithm may be attributed to the dispersion of initial population points in the search space. If the population does not cover the search space efficiently, it may not be able to locate the appropriate solution points, thereby missing the global optimum [44]. This difficulty may be minimized to a great extent by selecting a well-organized distribution law.

The PSO algorithm used to solve the optimization problem of the fuzzy predictive control uses the Gaussian distribution to initialize the swarm and is based on the ideas that take advantage of both prior knowledge about the search space

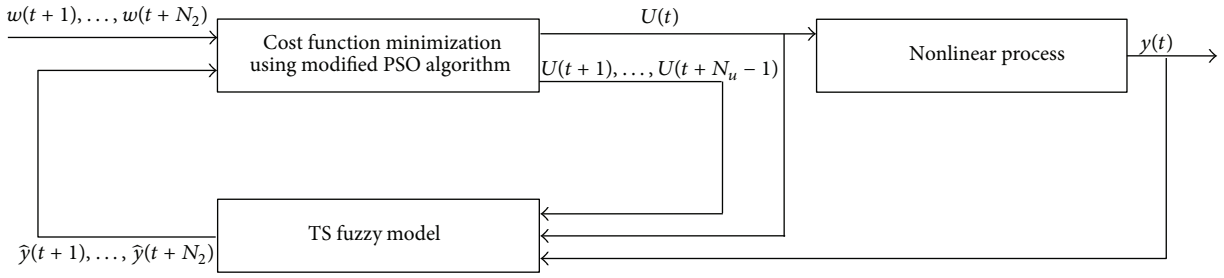

Figure 2: Block diagram of the proposed NMPC algorithm.

landscape and the fact that in most practical applications the dynamic optimization problems changes are gradual [19]. In most cases, it will be logical to assume that once a solution was found, it is better to track it rather than to look for it every time the optimization problem changes. In this algorithm a group of candidate solutions is generated and their fitness is evaluated. The fittest one is adopted by the PSO population as their initial global best position before even initializing the PSO algorithm; then the particles population is used to enhance this solution by favouring the search of an adaptive immediate neighbourhood around the initial solution. This allows increasing the efficiency of the algorithm in regard to the execution time and enhances the convergence quality of the solution. Contrary to the conventional PSO algorithms that involve the use of a large population size to effectively probe the whole search space, the downside, of course, is the huge computing requirement necessary to manage this population.

5. Fuzzy Predictive Controller Design Based on the PSO Algorithm

5.1. Notation and Principle. Without excluding the use of other nonlinear models, the TS fuzzy model described in [14] is used, as the explicit model to predict the future values of the system. In the case of other nonlinear models, the same steps can be used in solving the optimization problem.

Let us define the notation adopted in the formulation of the proposed control algorithm.

We consider a MIMO system with m inputs and n outputs.

The position of the hth particle, at the sampling time t, is represented by

$$X_h(t) = \left\{u^h(t), u^h(t+1), \ldots, u^h\left(t+N_u-1\right)\right\}, \quad h = 1, \ldots, N_{\text{pop}}, \tag{13}$$

where N_{pop} is the number of particles in the population (swarm), and

$$u^h(t+j) = \left[u_1^h(t+j), \ldots, u_m^h(t+j)\right]^T, \quad j = 0, \ldots, N_u - 1. \tag{14}$$

The position of each particle is adaptively changed according to (11) and (12).

The cost function J is used as the fitness function ($f_{\text{fitness}}(\cdot) = J(\cdot)$).

The control sequence $U(t)$ which is the optimal solution of the optimization problem is then given by the global best position ($X_{g\text{best}}$); that is,

$$U(t) = \left\{u(t), u(t+1), \ldots, u\left(t+N_u-1\right)\right\} = X_{g\text{best}}(t), \tag{15}$$

where $u(t+j) = [u_1(t+j), u_2(t+j), \ldots, u_m(t+j)]^T$, $j = 0, \ldots, N_u - 1$, is the system control input vector.

The block diagram of the developed control algorithm is illustrated in Figure 2.

The different steps of the algorithm are drawn based on the following points:

(1) The solution obtained at the sampling time t is taken as an initial solution for the next sampling time $t+1$:

$$X_1^0 = U(t). \tag{16}$$

This is a good choice when the system is operating in the steady state.

(2) Another candidate initial solution for the sampling time $t+1$ is given by

$$X_2^0 = \left\{u(t+1), u(t+2), \ldots, u\left(t+N_u\right)\right\}, \tag{17}$$

where $u(t+1), \ldots, u(t+N_u-1)$ is the control sequence obtained at the previous sampling time and $u(t+N_u) = u(t+N_u-1)$.

(3) Both initial solutions chosen in points 1 and 2 are evaluated using the fitness function. The fittest one is chosen as the initial global best position $X_{\text{best}}^0(t) = \{u^0(t), u^0(t+1), \ldots, u^0(t+N_u-1)\}$ of the entire PSO population.

(4) Using the Gaussian distribution, the particles ($X_h, h = 1, \ldots, N_{\text{pop}}$) are distributed around $X_{g\text{best}}^0$ within the radius $r_d = [r_{d1} \ \cdots \ r_{dm}]^T$ according to the following expression:

$$u_i^h(t+j) = u_i^0(t+j) + r_{di} y_j \frac{\sqrt[N_y]{k_m}}{\text{dist}_y} \quad \left(j = 0, \ldots, N_u - 1;\ i = 1, \ldots, m\right), \tag{18}$$

where y_j is random value from the normal distribution $N[0, 1]$, k_m is a random value from the normal distribution $N[0, 1/3]$, and $\text{dist}_y = \sqrt{\sum_{j=1}^{N_u} y_j^2}$.

The values of the radius r_d, at the sampling time $t + 1$, are given by

$$r_d = |u(t+1) - u(t)|. \tag{19}$$

To explore the whole search space, the particles positions are updated according to (11) and (12).

Using a Gaussian distribution will ensure that the density of the particles gets higher around the chosen initial global best position; thus the optimal solution can be reached in few iterations, using a small number of particles. This is very interesting, in the case of small changes of the system outputs, if the system is operating in the steady state, or if we want to fine-tune the solution. The use of a uniform distribution favors the diversity of the particles and increases the probability to find better solution relatively far from the previous one. Since the number of PSO particles and the allowed iterations number should not be large, their initial positions carry a lot of influence in the quality of the solution.

To keep the algorithm from converging toward a local optimum, a minimal value $r_{d\min} = [r_{d1\min} \cdots r_{dm\min}]^T$ given by the following expression is imposed to r_d:

$$r_{d\min} = \alpha_1 |w(t) - y(t)| + \alpha_2, \tag{20}$$

where α_1 is an $n \times m$ matrix of scaling parameters and α_2 ($m \times 1$) is used to impose a minimum value to $r_{d\min}$ regardless of the tracking error (the value of $r_{d\min}$ can have a fixed value by setting α_1 to zero).

If the current solution is far from the global optimum, the corresponding tracking error will be large and consequently the values of $r_{d\min}$ will also be large. When $r_{di\min} > r_{di}$ ($i = 1, \ldots, m$), the particles are distributed around $X^0_{g\text{best}}$ within $r_{di\min}$ instead of r_{di}. This will allow the algorithm to escape the current local optimum by looking for a better solution far from the actual one.

(5) The termination criterion is the maximal number of iterations.

5.2. Constraints Handling. One of the strong points of predictive control is its ability to handle the control constraints directly and efficiently in the design stage by solving a constrained optimization problem. Different types of constraints could be distinguished depending on their nature. Mainly, we have constraints on the inputs, presented by actuators with limited range of action and a limited slew rate, and constraints on the outputs due to environmental, safety, and economic goals.

Several approaches were proposed for handling constraints using evolutionary algorithms [45, 46]. They can be grouped into four categories [47]: methods based on preserving feasibility of solutions, methods based on penalty functions, methods that differentiate the feasible and infeasible solutions, and hybrid methods. The proposed control algorithm is based, in the constraints handling, on the hybrid approach. Indeed, the output constraints are handled using a penalty function while the input constraints are handled using the method based on preserving feasibility of solutions.

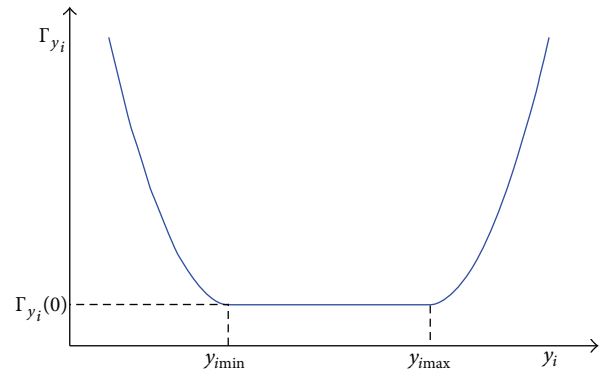

FIGURE 3: Weight function $\Gamma_{y_i}(y)$.

5.2.1. Outputs Constraints. One practical and popular approach is to soften the constraints [48]; this means that a violation of the constraints is allowed, but under severe and special conditions. The easy way to soften the outputs constraints is to add new variables, called slack variables, to the cost function that will heavily penalize any deviation or violation of the constraints in order to force the optimizer to violate these constraints only when it is absolutely necessary. The output-dependent weight function $\Gamma_y(y)$ was chosen to soften the output constraints. It has the following expression [5]:

$$\Gamma_y(y) = \begin{bmatrix} \Gamma_{y_1}(y_1) & 0 & \cdots & 0 \\ 0 & \Gamma_{y_2}(y_2) & \cdots & 0 \\ 0 & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & \Gamma_{y_n}(y_n) \end{bmatrix},$$

$$\Gamma_{y_i}(y_i) = \begin{cases} \Gamma_{y_i}(0)\left[1 + \varepsilon_i (y_i - y_{i\min})^2\right] & \text{if } y_i \le y_{i\min} \\ \Gamma_{y_i}(0) & \text{if } y_{i\min} \le y_i \le y_{i\max} \\ \Gamma_{y_i}(0)\left[1 + \varepsilon_i (y_i - y_{i\max})^2\right] & \text{if } y_i \ge y_{i\max}, \end{cases} \tag{21}$$

where $i = 1, \ldots, n$ and $y(t) = [y_1(t), y_2(t), \ldots, y_n(t)]^T$.

The function depicted in Figure 3 is the output-dependent weight function (Γ_{y_i}) used in the softening of the constraints.

ε_i is used to define the degree of softening: $\varepsilon_i = \infty$ indicates hard constraint while $\varepsilon_i = 0$ indicates no constraint. $y_{\min} = [y_{1\min}, \ldots, y_{n\min}]^T$ and $y_{\max} = [y_{1\max}, \ldots, y_{n\max}]^T$ limit the range where the outputs are allowed to exist.

The function Γ_{y_i} will be also used on unconstrained outputs (y_i) by setting $y_{i\min}$ and $y_{i\max}$, respectively, to $-\infty$ and $+\infty$.

By introducing the weight function $\Gamma_y(y)$, the optimization problem will be reformulated to have the following expression:

$$\begin{aligned}
&\min_{\Delta u(t)} \quad J\left(u(t), \widehat{y}(t), w(t)\right) \\
&\quad = \sum_{j=N_1}^{N_2} \left[\left(\widehat{y}(t+j \mid t) - w(t+j)\right)^T \Gamma_y \left(\widehat{y}(t+j \mid t) - w(t+j)\right)\right] + \sum_{j=1}^{N_u} \left[\Delta u^T (t+j-1) R \Delta u (t+j-1)\right] \\
&\text{subject to:} \quad \Delta u(t+j-1) = 0, \quad \text{for } j > N_u \\
&\qquad u_{\min} < u(t+j) < u_{\max} \quad \text{for } 1 \le j \le N_u \\
&\qquad \Delta u_{\min} < \Delta u(t+j-1) < \Delta u_{\max} \quad \text{for } j \le N_u.
\end{aligned} \tag{22}$$

5.2.2. Inputs Constraints. The PSO algorithm, as any other population based algorithms, can systematically handle the input constraints by searching for solutions in the space for which these constraints are satisfied. The lower and upper bounds of this space are obtained as follows.

The constraints on the inputs and their increments are typically given by

$$u_{\min} < u(t) < u_{\max} \tag{23}$$

$$\Delta u_{\min} < \Delta u(t) < \Delta u_{\max}, \tag{24}$$

where $u_{\min} = [u_{1\min}, \ldots, u_{m\min}]^T$ and $u_{\max} = [u_{1\max}, \ldots, u_{m\max}]^T$ limit the range of the inputs, while $\Delta u_{\min} = [\Delta u_{1\min}, \ldots, \Delta u_{m\min}]^T$ and $\Delta u_{\max} = [\Delta u_{1\max}, \ldots, \Delta u_{m\max}]^T$ limit the range of the inputs increments.

For any given particle h ($h = 1, \ldots, N_{\text{pop}}$), we can write, from (24),

$$\begin{aligned}
&\Delta u_{i\min} \le \Delta u_i^h (t+j \mid t) \le \Delta u_{i\max} \\
&\quad \Longrightarrow \Delta u_{i\min} + u_i^h (t+j-1 \mid t) \le u_i^h (t+j \mid t) \\
&\quad = u_i^h (t+j-1 \mid t) + \Delta u_i^h (t+j \mid t) \\
&\quad \le \Delta u_{i\max} + u_i^h (t+j-1 \mid t) \\
&\qquad (i = 1, \ldots, m; \; j = 0, \ldots, N_u - 1).
\end{aligned} \tag{25}$$

For $j = 0$, $u_i^h(t-1 \mid t)$ is the already calculated control action $u_i(t-1)$, and $\Delta u_i^h(t+j \mid t) = u_i^h(t+j \mid t) - u_i^h(t+j-1 \mid t)$.

On the other hand, from (23), the constraints on the inputs magnitude could also be written as follows:

$$\begin{aligned}
&u_{i\min} \le u_i^h (t+j \mid t) \le u_{i\max}, \\
&\qquad (i = 1, \ldots, m; \; j = 0, \ldots, N_u - 1).
\end{aligned} \tag{26}$$

Hence, from (25) and (26), the lower and upper bounds of the search space will be given by

$$\begin{aligned}
&L_{i\min}^h (t+j) \le u_i^h (t+j \mid t) \le L_{i\max}^h (t+j), \\
&\qquad (i = 1, \ldots, m; \; j = 0, \ldots, N_u - 1),
\end{aligned} \tag{27}$$

where

$$\begin{aligned}
L_{i\min}^h (t+j) &= \begin{cases} \Delta u_{i\min} + u_i^h (t+j-1 \mid t) & \text{if } \Delta u_{i\min} + u_i^h (t+j-1 \mid t) > u_{i\min} \\ u_{i\min} & \text{Otherwise,} \end{cases} \\
L_{i\max}^h (t+j) &= \begin{cases} \Delta u_{i\max} + u_i^h (t+j-1 \mid t) & \text{if } \Delta u_{i\max} + u_i^h (t+j-1 \mid t) < u_{i\max} \\ u_{i\max} & \text{Otherwise.} \end{cases}
\end{aligned} \tag{28}$$

5.3. The Control Algorithm. The basic steps of the proposed control algorithm are summarized as follows.

Step 1. Initialize the control algorithm by choosing suitable values for the design and the PSO algorithm parameters.

Step 2. According to points 1 and 2, determine the initial solutions X_1^0 and X_2^0.

Step 3. Specify the desired reference trajectory over the prediction horizon and compute the future values of the system outputs: $\widehat{y}(t+j)$, $N_1 \le j \le N_2$.

Step 4. Use the cost function (J) given by (22) to evaluate the fitness of the initial solutions X_1^0 and X_2^0. The fittest one will be adopted by the PSO population as their initial global best position $X_{g\text{best}}^0$.

Set $X_{g\text{best}} = X_{g\text{best}}^0$.

Step 5. Evaluate the value of r_d using (19) and the value of $r_{d\min}$ using (20):

$$\begin{aligned} &\text{If } r_{di} < r_{di\min} \text{ then} \\ &\quad r_{di} = r_{di\min}, \quad (i = 1, \ldots, m). \end{aligned} \tag{29}$$

Step 6. For each particle h,

randomly initialize its velocity v_h,

using (18), distribute the particle X_h around $X_{g\text{best}}$ within the radius r_d,

set $P_h = X_h$, and evaluate $f_{\text{fitness}}(P_h)$ (P_h is the personal best position of particle X_h)

$$\begin{aligned} &\text{If } f_{\text{fitness}}(P_h) < f_{\text{fitness}}\left(X_{g\text{best}}\right) \text{ then} \\ &\quad X_{g\text{best}} = P_h. \end{aligned} \tag{30}$$

Step 7 (repeat). For each particle h,

using (11) and (12), update X_h,

evaluate $f_{\text{fitness}}(X_h)$,

$$\begin{aligned} &\text{If } f_{\text{fitness}}(X_h) < f_{\text{fitness}}(P_h) \\ &\quad P_h = X_h \\ &\text{if } f_{\text{fitness}}(P_h) < f_{\text{fitness}}\left(X_{g\text{best}}\right) \text{ then} \\ &\quad X_{g\text{best}} = P_h, \end{aligned} \tag{31}$$

until termination criterion is reached.

Step 8. The optimal solution $U(t) = \{u(t), u(t+1) \cdots u(t+N_u-1)\}$ is the global best position $X_{g\text{best}}$.

For the next sampling time go to Step 2.

6. Case Study

6.1. Process Description. To assess the performance of the proposed control algorithm, the control of the industrial boiler model developed by Pellegrinetti and Bentsman [28] is considered. This model represents a dual fuel (oil and gas) boiler in the Abbott Power Plant in Champaign, Illinois, which could be used for both heating and electric generation. It is a MIMO process, which has four manipulated inputs (fuel flow rate, air flow rate, feed water flow rate, and a valve to regulate the steam demand) and four measured outputs (pressure, oxygen level in the flue gas, drum water level, and steam flow). The control objectives are to maintain the pressure, the level of the water in the drum, and the oxygen level in the flue gas at their desired values, regardless of the steam flow rate provided by the boiler and any other disturbances, such as the fluctuations in the heating value of the fuel or the variations in the ambient temperature. This mathematical model includes perturbations and measurement noises and is described by the following equations:

$$\begin{aligned} \dot{x}_1(t) &= c_{11}x_4(t)\,x_1^{9/8}(t) + c_{12}u_1(t-\tau_1) - c_{13}u_3(t-\tau_3) \\ &\quad + c_{14}, \\ \dot{x}_2(t) &= -c_{21}x_2(t) \\ &\quad + \frac{c_{22}u_2(t-\tau_2) - c_{23}u_1(t-\tau_1) - c_{24}u_1(t-\tau_1)\,x_2(t)}{c_{25}u_2(t-\tau_2) + c_{26}u_1(t-\tau_1)}, \\ \dot{x}_3(t) &= c_{31}x_1(t) - c_{32}x_4(t)\,x_1(t) + c_{33}u_3(t-\tau_3), \\ \dot{x}_4(t) &= -c_{41}x_4(t) + c_{42}u_1(t-\tau_1) + c_{43} + c_{41}u_4(t), \\ y_1(t) &= c_{51}x_1(t-\tau_4) + n_1(t), \\ y_2(t) &= c_{61}x_2(t-\tau_5) + n_2(t), \\ y_3(t) &= c_{70}x_1(t-\tau_6) + c_{71}x_3(t-\tau_6) \\ &\quad + c_{72}x_4(t-\tau_6)\,x_1(t-\tau_6) + c_{73}u_3(t-\tau_3-\tau_6) \\ &\quad + c_{74}u_1(t-\tau_1-\tau_6) \\ &\quad + \frac{\left[c_{75}x_1(t-\tau_6) + c_{76}\right]\left[1 - c_{77}x_3(t-\tau_6)\right]}{x_3(t-\tau_6)\left[x_1(t-\tau_6) + c_{78}\right]} + c_{79} \\ &\quad + n_3(t), \\ y_4(t) &= \left[c_{81}x_4(t-\tau_7) + c_{82}\right]x_1(t-\tau_7) + n_4(t), \end{aligned} \tag{32}$$

where $\dot{x}_1$ is the drum pressure state (Kgf/cm^2), y_1 is the measured drum pressure (PSI), y_2 and $\dot{x}_2$ are the measured excess oxygen level and its state, respectively (%), $\dot{x}_3$ is the system fluid density (Kg/m^3), y_3 is the drum water level (in), y_4 is the steam flow rate (Kg/s), u_1, u_2, and u_3 are, respectively, the fuel, air, and feed water flow rates which take values in the interval $[0 \;\; 1]$, and $\dot{x}_4$ is the exogenous variable related to the steam demanded. The variables n_i are the outputs of the first-order colored noise models driven by zero mean and unit variance white noise and are considered to be unmeasured output disturbances. The remaining coefficients values are given in Table 1. Consider

$$\begin{aligned} n_1 &= \frac{0.75s + 0.1}{s + 0.001}w_1, \\ n_2 &= \frac{0.019s + 0.001}{s + 0.024}w_2, \\ n_3 &= \frac{0.105s + 0.038}{s + 0.01}w_3, \\ n_4 &= \frac{0.01s + 0.0001}{s + 0.001}w_4, \end{aligned} \tag{33}$$

Table 1: Model equations coefficients.

$c_{11} = -0.00478$	$c_{31} = 0.00533176$	$c_{70} = -0.1048569$	
$c_{12} = 0.28$	$c_{32} = 0.025195$	$c_{71} = 0.15479$	$\tau_1 = 2$
$c_{13} = 0.01348$	$c_{33} = 0.7317058$	$c_{72} = 0.4954961$	$\tau_2 = 2$
$c_{14} = 0.02493$	$c_{41} = 0.04$	$c_{73} = -0.20797$	$\tau_3 = 3$
$c_{21} = 0.1540357$	$c_{42} = 0.0299886$	$c_{74} = 1.2720$	$\tau_4 = 3$
$c_{22} = 103.5462$	$c_{43} = 0.018088$	$c_{75} = -324212.7805$	$\tau_5 = 4$
$c_{23} = 107.4835$	$c_{51} = 14.214$	$c_{76} = -99556.24778$	$\tau_6 = 10$
$c_{24} = 1.9515$	$c_{61} = 1.00$	$c_{77} = 0.001185$	$\tau_7 = 2$
$c_{25} = 29.04$	$c_{81} = 0.85663$	$c_{78} = -1704.50476$	
$c_{26} = 1.824$	$c_{82} = -0.18128$	$c_{79} = -103.7351$	

where $n_i, i = 1, \ldots, 4$, are colored noise and $w_i, i = 1, \ldots, 4$, is the unit variance white noise.

The process is supposed to be initially operating around the following states:

$$x^0 = [22.5, 2.5, 621.17, 0.6941]^T,$$
$$y^0 = [320, 2.5, 0, 9.3053]^T, \quad (34)$$
$$u^0 = [0.3227, 0.39503, 0.37404, 0]^T.$$

The linear representation of the boiler around the operating state given by (34) is described by

$$\dot{x} = A_b x + B_b u,$$
$$y = C_b x + D_b u, \quad (35)$$

where

$$A_b = \begin{bmatrix} -0.005508 & 0 & 0 & -0.15872 \\ 0 & -0.20625 & 0 & 0 \\ -0.012156 & 0 & 0 & -0.566887 \\ 0 & 0 & 0 & -0.04 \end{bmatrix},$$
$$B_b = \begin{bmatrix} 0.28 & 0 & -0.01348 & 0 \\ -9.374893 & 7.658411 & 0 & 0 \\ 0 & 0 & 0.731706 & 0 \\ 0.029989 & 0 & 0 & 0.04 \end{bmatrix},$$
$$C_b = \begin{bmatrix} 14.214 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0.322072 & 0 & 0.149410 & 11.148662 \\ 0.413307 & 0 & 0 & 19.274175 \end{bmatrix}, \quad (36)$$
$$D_b = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1.272 & 0 & -0.20797 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

There is a direct feedthrough between the output y_3 and the inputs u_1 and u_3, expressed by a nonzero D matrix in the state representation. This property will create some problems when considering the linear MPC controller [49]. In order to have $D_b = 0$, a delay will be introduced to the inputs u_1 and u_3 with regard to the output y_3.

This process is inherently unstable. Essentially, a stabilisation scheme of the water level must be introduced to make any identification approaches possible by incorporating a proportional feedforward action (C2) of 0.0403 and a PI controller (C1) with $Kp = 0.258$ and $Ti = 1.1026e - 4$, as it is illustrated in Figure 4.

6.2. Process Fuzzy Identification. The first step in the design of a fuzzy NMPC controller is the construction of a nonlinear model of the process. Four fuzzy multi-inputs single output models were constructed to predict the future behaviour of the process. Two triangular membership functions per input, a sampling period of 3 s, and the models given by the following equations were used:

$$\hat{y}_1 = f_1\left(\hat{y}_1(k-1), u_1(k-5), \hat{x}_4(k-3)\right),$$
$$\hat{y}_3 = f_2\left(\hat{y}_3(k-1), u_1(k-12), r_{\text{Level}}(k-13), u_4(k-10), x_4(k-10)\right), \quad (37)$$
$$\hat{y}_4 = f_3\left(\hat{y}_4(k-1), u_1(k-4), u_4(k-2), x_4(k-2)\right),$$
$$\hat{x}_4 = f_4\left(\hat{x}_4(k-1), u_1(k-2), u_4(k-1)\right),$$

where f_i, $i = 1, \ldots, 4$, represent the fuzzy relationships between the inputs and the outputs.

Figure 5 shows the validation results of these models.

6.3. Controller Implementation and Simulation Results. It has been shown in [28] that there is a nearly linear relationship between the fuel/air ratio (FAR $= u_2/u_1$) and the oxygen level. So, the oxygen level y_2 can be kept constant at a desired value by maintaining the corresponding FAR value constant. Given that u_1 affects all the systems outputs, varying u_2 to maintain the FAR value constant is the appropriate way. Thus, the value of u_2 can be directly deduced from that of u_1. To keep the desired oxygen level in flue gas y_2 around 2.5% the FAR value must be equal to 1.2241 and u_1 equals 0.8169 u_2.

The control objective is to force the steam flow rate y_4 to track a desired reference trajectory, while fulfilling the following constraints on the manipulated inputs,

$$\Delta u_{1\,\text{Max}} = 0.1,$$
$$\Delta u_{2\,\text{Max}} = 0.1,$$
$$\Delta u_{r_{\text{Level}}\,\text{Max}} = 1 \quad -4 < r_{\text{Level}} < 4, \quad (38)$$
$$\Delta u_{4\,\text{Max}} = 0.1 \quad -0.2 < u_4 < 0.35,$$
$$0 < u_3 < 1,$$

and the following constraints on the outputs,

$$y_1 = 320,$$
$$y_2 = 2.5, \quad (39)$$
$$y_3 = 0.$$

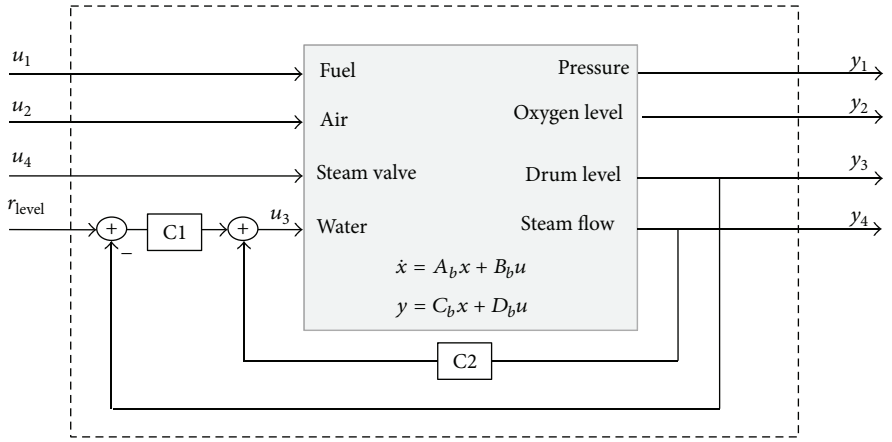

Figure 4: Stabilization scheme for the boiler.

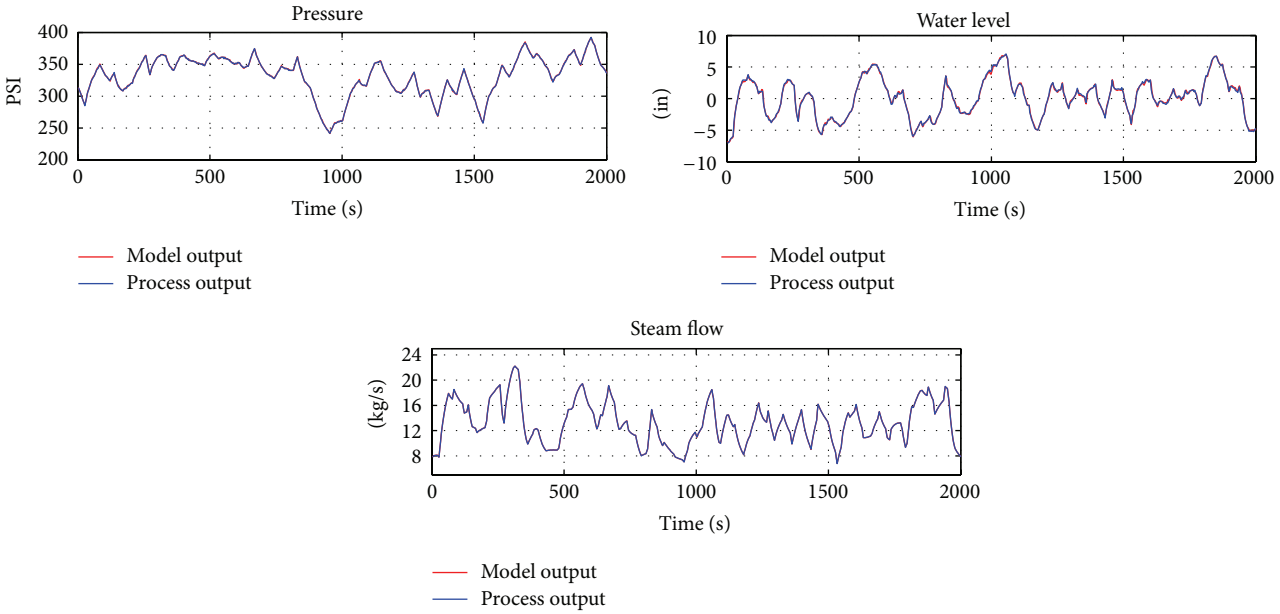

Figure 5: Validations of the fuzzy models.

The following parameters values were used in implementing the proposed control algorithm:

$$R = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 130 \end{bmatrix}, \quad \alpha_1 = \begin{bmatrix} 1.65e-4 & 0 & 9.54e-5 & 3e-3 \\ 0 & 0 & 0 & 0 \\ 1.65e-3 & 0 & 1.05e-2 & 2.76e-2 \\ 1.28e-4 & 0 & 1.246e-4 & 3e-3 \end{bmatrix}. \tag{40}$$

Table 2: Design parameters.

	N_1	N_2	ε_i	$\Gamma(0)$	$y_{\min}$	$y_{\max}$
$\hat{y}_1$	1	10	1	0.0085	318	322
$\hat{y}_3$	10	20	0.7	0.0065	−1.5	1.5
$\hat{y}_4$	1	10	0	1.006	/	/
$\hat{x}_4$	1	10	0	0	/	/

And $N_u = 2$, number of particles = 5, number of iterations = 5, and $\alpha_2 = [0 \ 0 \ 0 \ 0]^T$. The values of the other parameters are gathered in Table 2.

In addition to the proposed algorithm, two other control strategies were considered:

Using the Conventional PSO. In this algorithm, the particles are randomly distributed in the whole search space and the

solution of the previous sampling time is taken as a possible solution for the current sampling time. The PSO parameters values $\chi = 0.729843788$ and $c = 2.05$, adopted by Clerc and Kennedy [34], are used in both PSO based controllers.

Using the Linear MPC Strategy. In this strategy, only the constraints on the inputs variables and their increments are considered and the optimization problem is converted into a quadratic programming (QP) problem.

To implement this strategy, a linear representation for the process given in Figure 4 must be used. To remove the direct feedthrough between the output y_3 and inputs u_1 and u_3, the structure given in Figure 4 is slightly modified by introducing a unit delay in these inputs (Figure 6). The obtained system is linearized around the operating state given by (34) and then discretized using a sampling period of 3 s. The following discrete state space representation is obtained:

$$\begin{aligned} \dot{x}(k+1) &= Ax(k) + Bu(k), \\ y(k+1) &= Cx(k), \end{aligned} \tag{41}$$

where

$$A = \begin{bmatrix} 0.9861 & 0 & 0.001485 & -0.3706 \\ 0 & 0.5386 & 0 & 0 \\ -0.1736 & 0 & 0.9187 & -5.665 \\ 0 & 0 & 0 & 0.8869 \end{bmatrix},$$

$$B = \begin{bmatrix} 0.8172 & 0 & -0.009937 & -0.02264 \\ -20.97 & 17.13 & 0 & 0 \\ -0.3384 & 0 & 0.5441 & -0.3525 \\ 0.08478 & 0 & 0 & 0.1131 \end{bmatrix}, \tag{42}$$

$$C = \begin{bmatrix} 14.21 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0.3221 & 0 & 0.1494 & 11.15 \\ 0.4133 & 0 & 0 & 19.27 \end{bmatrix}.$$

The design parameters values are chosen as follows:

$$R = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 130 \end{bmatrix},$$

$$Q = \begin{bmatrix} 0.4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 2.006 \end{bmatrix}. \tag{43}$$

And $N_1 = 1$, $N_2 = 15$, and $N_u = 2$.

Figure 7 presents the obtained results when a population size of 5 particles and 5 iterations is used. A zoom of the area lying between the 17 and 35 minutes of the steam flow rate is given by Figure 8. The corresponding control signals are depicted in Figure 9.

The obtained results show that the proposed controller outperforms the linear MPC and the conventional PSO based controller. The output y_2 is almost identical for the three controllers, due to the fact that the FAR is always kept constant regardless of the used controller.

To compare the performance of the considered algorithms when different values of the population size and the number of iterations are used, the Mean Cost Value (MCV) criterion given by (44) is adopted:

$$\text{MCV} = \frac{1}{N}\sum_{t=1}^{N} J(U(t)), \tag{44}$$

N is the number of samples.

For each value of the population size and the number of iterations both control algorithms are executed fifteen times and the corresponding value of the MCV criterion is computed. The averages values of the MCV criterion, over the fifteen executions, corresponding to each population size and iterations number are depicted in Figure 10. It is clear, from this figure, that the proposed PSO based controller always outperforms the conventional PSO based controller, regardless of the selected population size and the number of iterations.

The execution time of a control algorithm is a very important parameter to evaluate its computing efficiency and real-time implementation feasibility. Table 3 gives the execution time of the considered control algorithms for different population size and number of iterations; they were executed on an Intel Core i5 3.10 GHz based machine.

Table 3 indicates that the execution time of the proposed algorithm is slightly higher than that of the conventional algorithm. Although, when the population size or the iteration number increases, the execution times of both algorithms get closer. It was expected to get slightly higher computing time with the proposed controller as it requires more operations than the conventional PSO based controller. However, this small increase in the computation requirement has produced a decent performance increase.

Table 3: Execution time of the considered algorithms.

Configuration (population × iteration)	Proposed PSO based controller (ms)	Conventional PSO based controller (ms)
5 × 5	223.26	207.86
5 × 10	398.42	385.98
5 × 15	572.94	560.08
5 × 20	746.91	736.00
5 × 25	922.58	916.40
10 × 10	771.95	768.84

7. Conclusion

A particle swarm optimization algorithm that uses the Gaussian distribution was proposed to solve the constrained

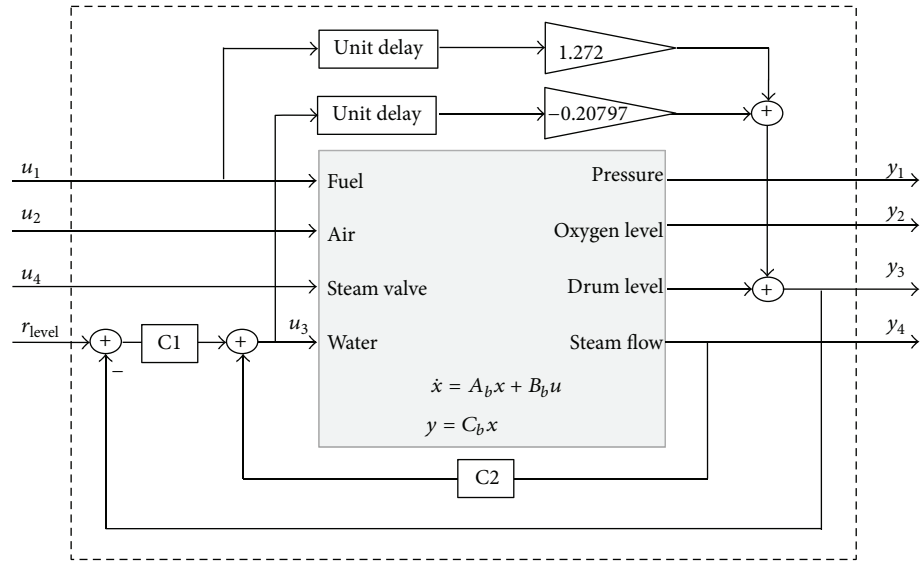

Figure 6: Stabilization scheme for the boiler without direct feedthrough.

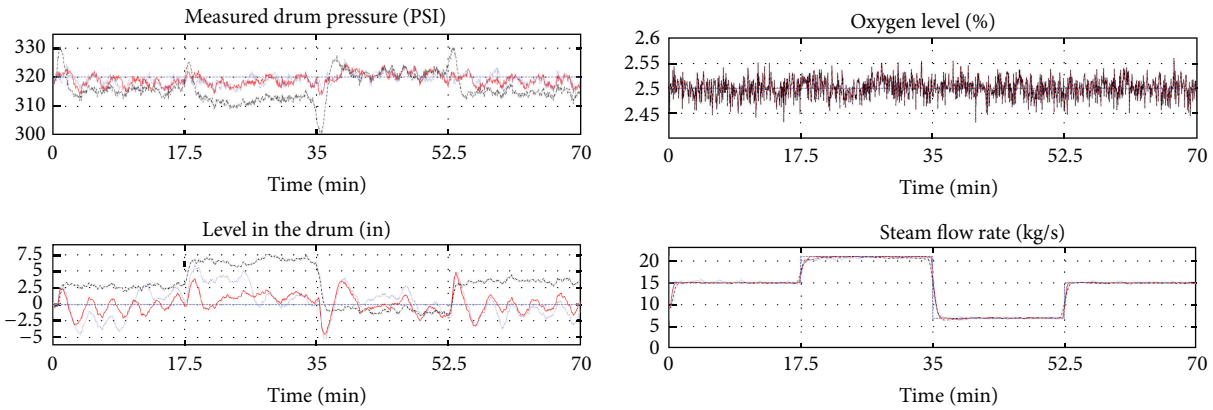

Figure 7: Response of the boiler using the three controllers (red solid line: proposed controller; blue dotted line: conventional PSO based controller; black dashed dotted line: linear MPC; and blue dashed line: reference trajectory).

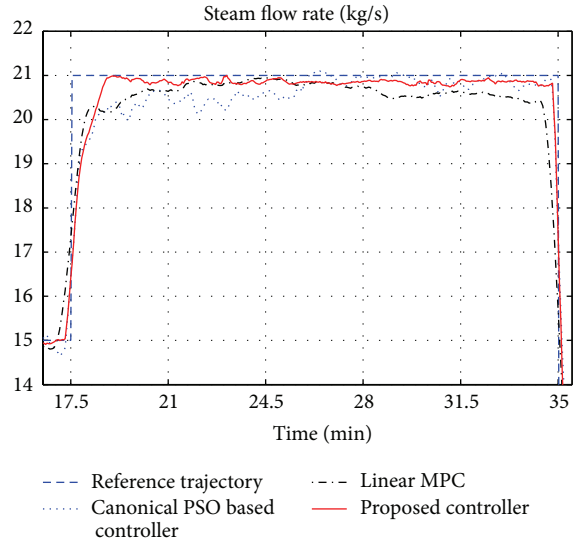

Figure 8: Zoomed area between the instants 17.5 and 35 min of the steam flow rate.

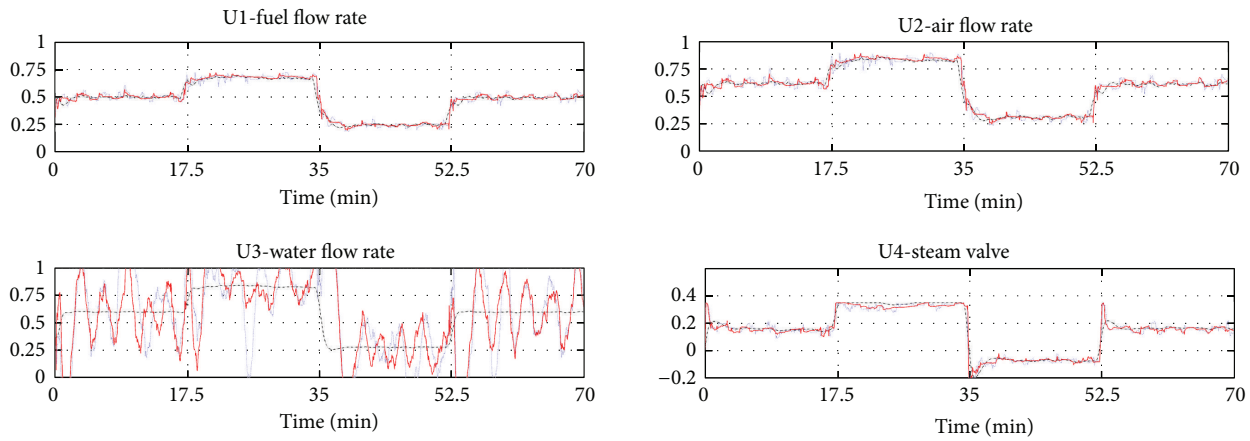

Figure 9: Control signals for the three controllers (red solid line: proposed controller; blue dotted line: conventional PSO based controller; and black dashed dotted line: linear MPC).

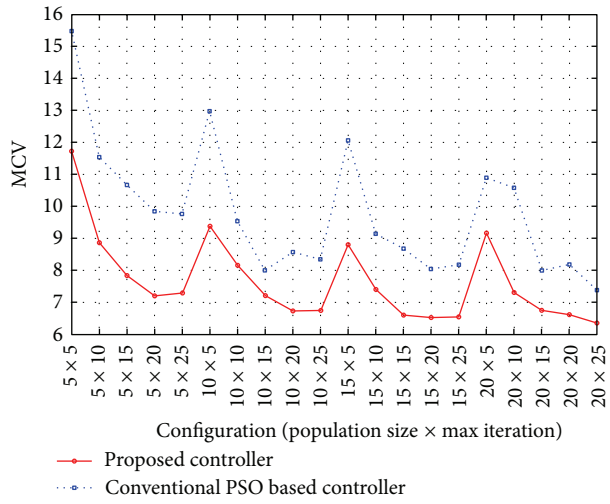

Figure 10: MCV average values.

fuzzy predictive control problem. A Takagi-Sugeno fuzzy model was used to predict the future values of the system outputs. Fuzzy systems are known for their simplicity and their ability to approximate complex nonlinear processes. However, when using a nonlinear model the optimization problem to be solved is then a nonlinear and a nonconvex one. The solution of this problem using numerical methods is expensive in computational time, especially when considering constraints. Particle swarm optimization algorithms are powerful metaheuristic based algorithms that require much less computational effort than the genetic algorithms, and their implementation is relatively easy. In this work, the PSO algorithm was adapted to find an efficient solution to the fuzzy predictive control optimization problem with a low computational burden. This was done essentially by reducing both the PSO population size and the number of iterations needed to obtain a suitable solution.

The efficiency and the control accuracy of the proposed algorithm were investigated and compared to other control strategies by considering the control of a highly nonlinear MIMO process, namely, an industrial boiler. The obtained simulation results have proved the efficiency and the control accuracy of the proposed algorithm, when considering highly nonlinear systems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] S. D. Cairano, D. Bernardini, A. Bemporad, and I. V. Kolmanovsky, "Stochastic MPC with learning for driver-predictive vehicle control and its application to HEV energy management," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 3, pp. 1018–1031, 2014.

[2] M. A. F. Martins, A. C. Zanin, and D. Odloak, "Robust model predictive control of an industrial partial combustion

fluidized-bed catalytic cracking converter," *Chemical Engineering Research and Design*, vol. 92, no. 5, pp. 917–930, 2014.

[3] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003.

[4] E. F. Camacho and C. Bordons, *Model Predictive Control*, Springer, London, UK, 1999.

[5] B. Kouvaritakis and M. Cannon, *Nonlinear Predictive Control, Theory and Practice*, The Institution of Engineering and Technology, Stevenage, UK, 2001.

[6] J. Richalet and D. O'Donovan, *Predictive Functional Control, Principles and Industrial Applications*, Springer, London, UK, 2009.

[7] H. Sarimveis and G. Bafas, "Fuzzy model predictive control of non-linear processes using genetic algorithms," *Fuzzy Sets and Systems*, vol. 139, no. 1, pp. 59–80, 2003.

[8] P. M. Marusak, "Advantages of an easy to design fuzzy predictive algorithm in control systems of nonlinear chemical reactors," *Applied Soft Computing Journal*, vol. 9, no. 3, pp. 1111–1125, 2009.

[9] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[10] H.-G. Han, X.-L. Wu, and J.-F. Qiao, "Real-time model predictive control using a self-organizing neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 9, pp. 1425–1436, 2013.

[11] M. Khairy, A. L. Elshafei, and H. M. Emara, "LMI based design of constrained fuzzy predictive control," *Fuzzy Sets and Systems*, vol. 161, no. 6, pp. 893–918, 2010.

[12] B. Kosko, "Fuzzy systems as universal approximators," in *Proceedings of the 1st IEEE International Conference on Fuzzy Systems (FUZZ '92)*, pp. 1153–1162, March 1992.

[13] B. Ding and X. Ping, "Output feedback predictive control with one free control move for nonlinear systems represented by a takagi-sugeno model," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 2, pp. 249–263, 2014.

[14] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 15, no. 1, pp. 116–132, 1985.

[15] H. Jiang, C. K. Kwong, Z. Chen, and Y. C. Ysim, "Chaos particle swarm optimization and T-S fuzzy modeling approaches to constrained predictive control," *Expert Systems with Applications*, vol. 39, no. 1, pp. 194–201, 2012.

[16] J. Espinosa, J. Vandewalle, and V. Wertz, *Fuzzy Logic, Identification and Predictive Control*, Advances in Industrial Control, Springer, London, UK, 2005.

[17] M. Herceg, M. Kvasnica, and M. Fikar, "Parametric approach to nonlinear model predictive control," in *Nonlinear Model Predictive Control*, L. Magni, D. M. Raimondo, and F. Allgöwer, Eds., vol. 384 of *Lecture Notes in Control and Information Sciences*, pp. 381–389, Springer, Berlin, Germany, 2009.

[18] J. A. Roubos, R. Babuska, P. M. Bruijn, and H. B. Verbruggen, "Predictive control by local linearization of a Takagi-Sugeno fuzzy model," in *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp. 37–42, May 1998.

[19] C. Blum and D. Merkle, *Swarm Intelligence, Introduction and Applications*, Springer, Berlin, Germany, 2008.

[20] J. P. Coelho, P. B. D. M. Oliveira, and J. B. Cunha, "Greenhouse air temperature predictive control using the particle swarm optimisation algorithm," *Computers and Electronics in Agriculture*, vol. 49, no. 3, pp. 330–344, 2005.

[21] Z. G. H. Hou and H. Li, "Neural networks predictive control using AEPSO," in *Proceedings of the 27th Chinese Control Conference*, pp. 180–183, 2008.

[22] Y. Song, Z. Chen, and Z. Yuan, "New chaotic PSO-based neural network predictive control for nonlinear process," *IEEE Transactions on Neural Networks*, vol. 18, no. 2, pp. 595–600, 2007.

[23] X. Wang and J. Xiao, "PSO-based model predictive control for nonlinear processes," in *Advances in Natural Computation*, L. Wang, K. Chen, and Y. Ong, Eds., vol. 3611, pp. 196–203, Springer, Berlin, Germany, 2005.

[24] W. H. Lim and N. A. M. Isa, "Particle swarm optimization with increasing topology connectivity," *Engineering Applications of Artificial Intelligence*, vol. 27, pp. 80–102, 2014.

[25] H. Wang, H. Sun, C. Li, S. Rahnamayan, and J.-S. Pan, "Diversity enhanced particle swarm optimization with neighborhood search," *Information Sciences*, vol. 223, pp. 119–135, 2013.

[26] J. Ding, J. Liu, K. R. Chowdhury, W. Zhang, Q. Hu, and J. Lei, "A particle swarm optimization using local stochastic search and enhancing diversity for continuous optimization," *Neurocomputing*, vol. 137, pp. 261–267, 2014.

[27] J. Espinosa and J. Vandewalle, "Predictive control using fuzzy models applied to a steam generating unit," in *Proceedings of the 3rd International Workshop on Fuzzy Logic and Intelligent Technologies for Nuclear Science and Industry*, 1998.

[28] G. Pellegrinetti and J. Bentsman, "Nonlinear control oriented boiler modeling—a benchmark problem for controller design," *IEEE Transactions on Control Systems Technology*, vol. 4, no. 1, pp. 57–64, 1996.

[29] P. Kittisupakorn, P. Thitiyasook, M. A. Hussain, and W. Daosud, "Neural network based model predictive control for a steel pickling process," *Journal of Process Control*, vol. 19, no. 4, pp. 579–590, 2009.

[30] A. Schäfer, P. Kühl, M. Diehl, J. Schlöder, and H. G. Bock, "Fast reduced multiple shooting methods for nonlinear model predictive control," *Chemical Engineering and Processing: Process Intensification*, vol. 46, no. 11, pp. 1200–1214, 2007.

[31] A. Steinboeck, D. Wild, and A. Kugi, "Nonlinear model predictive control of a continuous slab reheating furnace," *Control Engineering Practice*, vol. 21, no. 4, pp. 495–508, 2013.

[32] Y. Li, J. Shen, K. Y. Lee, and X. Liu, "Offset-free fuzzy model predictive control of a boiler-turbine system based on genetic algorithm," *Simulation Modelling Practice and Theory*, vol. 26, pp. 77–95, 2012.

[33] Y. L. Huang, H. H. Lou, J. P. Gong, and T. F. Edgar, "Fuzzy model predictive control," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 6, pp. 665–678, 2000.

[34] M. Clerc and J. Kennedy, "The particle swarm—explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[35] M. Clerc, *Particle Swarm Optimization*, ISTE, London, UK, 2006.

[36] L. Lanzarini, V. Leza, and A. De Giusti, "Particle swarm optimization with variable population size," in *Artificial Intelligence and Soft Computing—ICAISC 2008*, L. Rutkowski, R. Tadeusiewicz, L. Zadeh, and J. Zurada, Eds., vol. 5097 of *Lecture Notes in Computer Science*, pp. 438–449, Springer, Berlin, Germany, 2008.

[37] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: philosophy and performance differences," in *Proceedings of the 7th Annual Conference on Evolutionary*

Programming VII (EP '98), pp. 601–610, San Diego, Calif, USA, March 1998.

[38] Y. Song, Z. Chen, and Z. Yuan, "New chaotic PSO-based neural network predictive control for nonlinear process," *IEEE Transactions on Neural Networks*, vol. 18, no. 2, pp. 595–601, 2007.

[39] K. T. Chaturvedi, M. Pandit, and L. Srivastava, "Self-organizing hierarchical particle swarm optimization for nonconvex economic dispatch," *IEEE Transactions on Power Systems*, vol. 23, no. 3, pp. 1079–1087, 2008.

[40] R. A. Krohling, "Gaussian swarm: a novel particle swarm optimization algorithm," in *Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems*, pp. 372–376, December 2004.

[41] R. A. Krohling and L. D. S. Coelho, "PSO-E: particle swarm with exponential distribution," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 1428–1433, July 2006.

[42] M. Pant, T. Radha, and V. P. Singh, "Particle swarm optimization: experimenting the distributions of random numbers," in *Proceedings of the 3rd Indian International Conference on Artificial Intelligence (IICAI '07)*, pp. 412–420, 2007.

[43] D. K. Gehlhaar and D. B. Fogel, "Tuning evolutionary programming for conformationally flexible molecular docking," in *Proceedings of the 5th Annual Conference on Evolutionary Programming*, pp. 419–429, 1996.

[44] C. Grosan, A. Abraham, and M. Nicoara, "Search optimization using hybrid particle sub-swarms and evolutionary algorithms," *International Journal of Simulation Systems, Science & Technology*, vol. 6, pp. 60–79, 2005.

[45] C. A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11-12, pp. 1245–1287, 2002.

[46] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1–32, 1996.

[47] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *Evolutionary Computation*, vol. 7, no. 1, pp. 19–44, 1999.

[48] J. A. Rossiter, *Model-Based Predictive Control: A Practical Approach*, CRC Press, Boca Raton, Fla, USA, 2004.

[49] J. M. Maciejowski, *Predictive Control with Constraints*, Prentice Hall, 2002.