International Journal of Innovative Computing, Information and Control
Volume **9**, Number **9**, September **2013**

ICIC International ©2013 ISSN 1349-4198
pp. 3741–3758

A NEW APPROACH FOR FUZZY PREDICTIVE ADAPTIVE CONTROLLER DESIGN USING PARTICLE SWARM OPTIMIZATION ALGORITHM

Sofiane Bououden[1], Mohammed Chadli[2], Fouad Allouani[1] and Salim Filali[3]

[1]Institut des Sciences et Technologie
Université Abbes Laghrour
Route de Constantine BP 1252, El Houria, 40004 Khenchela, Algérie
ss_bououden@yahoo.fr

[2]University of Picardie Jules Verne, MIS (EA 4029)
33, rue Saint-Leu, 80039 Amiens, France
mchadli@u-picardie.fr

[3]Laboratoire d'Automatique et de Robotique
Université de Constantine
Route d'Ain El Bey, Constantine, Algérie

Received June 2012; revised December 2012

Abstract. *This paper introduces a new approach for designing an adaptive fuzzy model predictive control (AFMPC) using the Particle Swarm Optimization (PSO) algorithm. The system to be controlled is modeled by a Takagi-Sugeno fuzzy inference system whose parameters are identified using recursive least square algorithm. These parameters are used to calculate the objective function based on predictive approach and structure of RST controller. The controller design methodology is formulated as an optimization problem solved by PSO algorithm to obtain the optimal future control. The approach was applied for controlling two non linear systems CSTR and Tank system. The results are encouraging compared with those obtained using the Proportional Integral-Particle Swarm Optimization (PI-PSO) and adaptive fuzzy model predictive control (AFMPC).*

Keywords: Adaptive fuzzy logic systems, Predictive control, Particle swarm optimization (PSO)

1. Introduction.

Model predictive control (MPC) has been an active field of research during the last three decades, driven by numerous successful applications of the technology [1-4]. Most industrial plants exhibit a substantial level of nonlinearity and uncertainty. They present difficulties from the perspective of control engineering and the design of controllers. However, the continuous and batch processes in chemical and petro-chemical plants are nonlinear where some of these plants show a high degree of nonlinearity. For a highly nonlinear system, MPC algorithm does not give satisfactory dynamic performance.

However, the fuzzy models of the Takagi-Sugeno (T-S) type proved to be suitable for the use in nonlinear MPC, because of their ability to give an accurate approximation of the complex nonlinear systems. This can be done by combining the data with the prior knowledge [5-9]. An adaptive fuzzy logic systems (AFLS) is employed to determine the controller structure as well as the free parameters of the adaptive fuzzy logic systems [10-15]. In the AFMPC, the AFLS is used as the prediction model of the nonlinear plant and the system performance is greatly dependent upon the online optimization. Recently, several algorithms have been found to be effective in control of a wide class of nonlinear process [16,17]. However, most previous works usually provide local optima, require the AFMPC cost function differential, and they are still a complex procedure for calculating the Jacobian and inverse Hessian matrix at each sampling step which is difficult to achieve in real time even under some simplifications. Another reason, a major problem in nonlinear programming when the function optimized is highly non-convex, which will in general have several local minima [18]. Due to the mentioned difficulties related to AFMPC and nonlinear programming, the intelligent evolutionary algorithms are more suitable for the optimizing in AFMPC [19-21]. The particle swarm optimization (PSO) is recently proved to be successful approach to solve complex optimization problem [22]. This algorithm iteratively explores a multidimensional search space with a swarm of individuals (referred to as ‘particles’), looking for the global optima (minimum or maximum) [23-27].

This paper proposes improved PSO which combines an AFMPC approach, called PSO-based AFMPC, to obtain the optimal future control input for predictive control algorithm. Firstly, the fuzzy logic is used to identify the parameters of the subsystems. These parameters are updated according to a recursive adaptation, which allows to calculate the predictive control. The second step can be done by transforming the control law to a polynomial form [28], which should be employed by the PSO algorithm for parameter optimization problem of RST controller. The main idea is based on the minimization of the predictive cost function maximum. This choice might allow the PSO algorithm to attain rapidly toward the best optimal RST controller. In addition, one of the advantages of this algorithm is to avoid the inversion matrices when calculating the predictive control law, which is difficult to achieve in a real time. The proposed PSO-based AFMPC method is tested on two nonlinear processes. Numerical results are compared with the other hybrid PSO methods and non PSO methods available in the usual literature. The performance study demonstrates the effectiveness and efficiency of the proposed PSO-based AFMPC approach.

The work is structured as follows. In Section 2, the nonlinear system is modeled by Takagi-Sugeno fuzzy modeling. Section 3 presents the predictive control in his RST polynomial form. In Section 4, the RST fuzzy adaptive control is coupled with the PSO algorithm to get the PSO-based AFMPC structure which optimizes the controller parameters. In order to show the good performance of the proposal approach, simulation results are given in Section 5. Finally, Section 6 concludes.

2. T-S Fuzzy Modeling.

Consider a single-input single-output (SISO) nonlinear system. The system is decomposed into ‘r’ subsystems such that each subsystem demonstrates a linear or nearly linear behavior. Using the Takagi-Sugeno’s modeling methodology [5,6,29,30], a fuzzy quasi-linear model, R^i or fuzzy implication, is developed for each subsystem. In such a model, the cause-effect relationship between control u and output y at sampling time k is established in a discrete time representation. The subsystems are defined in the fuzzy regions, R^i, $i = 1, 2, \dots, r$.

A SISO discrete-time nonlinear system can be described as follows:

$$\begin{aligned} x(k+1) &= F(x(k), u(k)) \\ y(k) &= H(x(k)) \end{aligned} \tag{1}$$

where $u(k) \in R$ and $y(k) \in R$ are the system input and output at time k respectively, $x(k) \in R^n$ is the state vector of the system, and $F(x(k), u(k)) \in R^n$ and $H(x(k)) \in R$ are nonlinear functions.

It is assumed that $F(0, 0) = 0$ and $H(0) = 0$. For both a controllable and observable system, $x(k)$ can be expressed as function of $y(k), \dots, y(k-n+1), u(k), \dots, u(k-n+1)$, and n represents the order of the system. Therefore, when the nonlinear system (1) is investigated around the origin, its equivalent system can be expressed as follows [31]:

$$\begin{aligned} y(k+d) &= a_1 y(k) + \ldots + a_{na} y(k-na-1) + b_0 u(k) + \ldots + b_{nb} u(k-nb) + l \\ &= \xi(k)^T \theta + l \end{aligned} \tag{2}$$

where $1 \leq d < n$ corresponds to the time delay of the system, and $\xi(k) = [y(k), y(k-1), \ldots, y(k-n_a-1), u(k), \ldots, u(k-n_b)]^T$ is referred to as the regression vector $\theta = [a_1, \ldots, a_{na}, b_0, \ldots, b_{nb}]^T$. When the time delay is $d = 1$, system (2) can be rewritten as Controlled Auto-Regressive Integrated Moving Average model (CARIMA) [28]:

$$A(z^{-1})y(k) = B(z^{-1})u(k-1) + \frac{\zeta(t)}{\Delta(z^{-1})} \tag{3}$$

where $A(z^{-1})$ and $B(z^{-1})$ are polynomials in the backward shift operator z^{-1}

$$\begin{cases} A(z^{-1}) = 1 + a_1 z^{-1} + \ldots + a_{na} z^{-na} \\ B(z^{-1}) = b_0 + b_1 z^{-1} + \ldots + b_{nb} z^{-nb} \end{cases}$$

$\zeta(t)$ is an uncorrelated random sequence and the use of the operator $\Delta(z^{-1}) = 1 - z^{-1}$ ensures an integral control law or a closed loop type I system.

A fuzzy implication (FI) is rule based on and consists of a set of symbolic antecedents in the IF part (premise) and a linear numerical expression in the THEN part (consequence). Using a CARIMA model structure, the fuzzy implication of the (3) can be written as follows [13]:

$$\begin{aligned} R^i : \ & \text{IF } y(k) \text{ is } M_1^i, y(k-1) \text{ is } M_2^i, \ldots, y(k-na-1) \text{ is } M_{na}^i \text{ and } u(k) \text{ is } L_0^i, \\ & u(k-1) \text{ is } L_1^i, \ldots, u(k-nb) \text{ is } L_{nb}^i \\ & \text{THEN } y^i(k+1) = \frac{B^i(z^{-1})}{A^i(z^{-1})} u(k), \quad i = 1, \ldots, r \end{aligned} \tag{4}$$

where M_j^i fuzzy set is corresponding to output $y(k-j)$ in the ith FI; L_p^i fuzzy set is corresponding to output $u(k-p)$ in the ith FI;

The system output $y(k+1)$ is computed as the weighted average of the individual rules consequents

$$y(k+1) = \frac{\sum_{i=1}^{r} w^i u(k) B^i(z^{-1})/A^i(z^{-1})}{\sum_{i=1}^{r} w^i} \tag{5}$$

The degree of fulfillment of the ith rule, w^i is obtained as the product of the membership degrees of the antecedent variables in that rule

$$w^i = \prod_{j=1}^{na} \mu_{M_j^i}(y(k-j-1)) \prod_{p=0}^{nb} \mu_{M_j^i}(u(k-p)) \tag{6}$$

In this paper, the nonlinear system is approximated by the AFLS, where the adaptation parameters are obtained using the recursive least square (RLS) algorithm. This algorithm is stopped when the value of ε is lower than the proposed error between the actual output and estimated output. For more details see [14]. The identification system allows calculating the predictive control law RST, which will be applied on the nonlinear system.

3. **Predictive Control.**

This part represents the developments of the predictive control [1,32], where it presents the prediction model under the input/output form obtained by

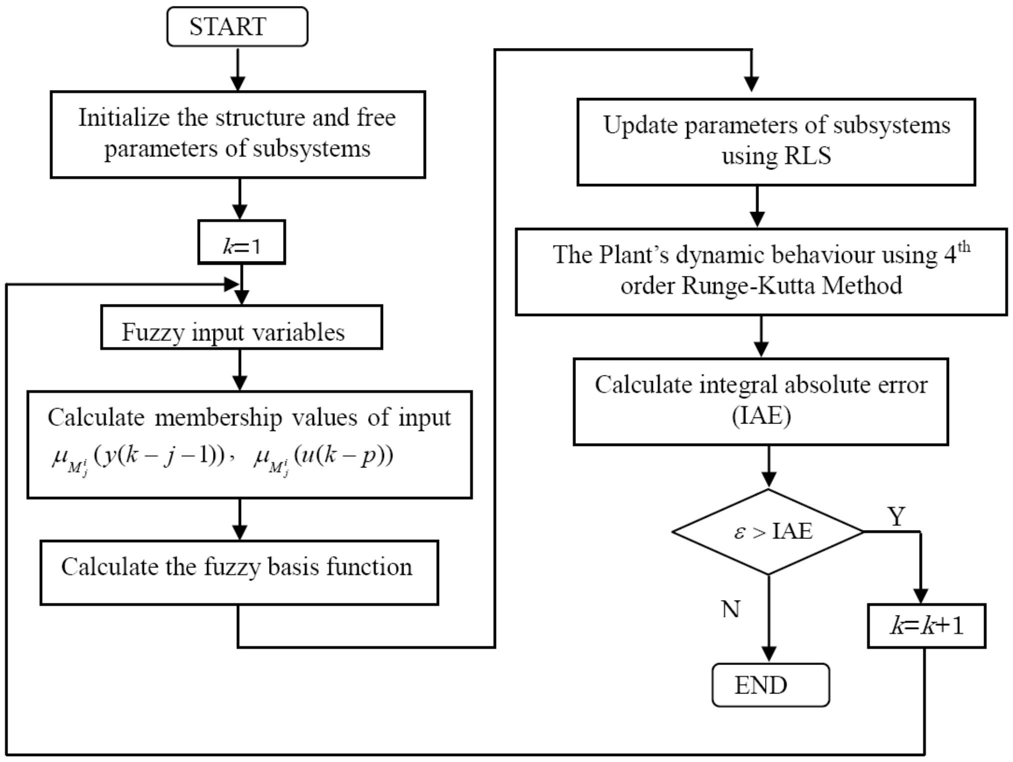

START
Initialize the structure and free parameters of subsystems
k=1
Fuzzy input variables
Calculate membership values of input $\mu_{M_j^i}(y(k-j-1))$, $\mu_{M_j^i}(u(k-p))$
Calculate the fuzzy basis function
Update parameters of subsystems using RLS
The Plant's dynamic behaviour using 4th order Runge-Kutta Method
Calculate integral absolute error (IAE)
ε > IAE
Y
N
k=k+1
END

Figure 1. Flowchart representation of AFLS algorithm. k: plant simulation counter. IAE: integral of the absolute value of error.

fuzzy adaptive model (5). From the previous model (5), a polynomial optimal predictor is designed under the following form [1,28]:

$$\hat{y}(t+j) = \underbrace{F_j(z^{-1})y(t) + H_j(z^{-1})\Delta u(t-1)}_{free\ response} + \underbrace{G_j(z^{-1})\Delta u(t+j-1) + J_j(z^{-1})\zeta(t+j)}_{forced\ response} \tag{7}$$

where the unknown polynomials F_j, G_j, H_j and J_j are derived by solving the Diophantine equation. In the further developments, the term related to future disturbances, corresponding to the prediction of $\xi(t+j)$, is set to zero. The criterion is a weighted sum of square predicted future errors and square control signal increments:

$$J = \sum_{j=N_1}^{N_2} (\hat{y}(t+j) - w(k+j))^2 + \lambda \sum_{j=N_1}^{Nu} \Delta u(t+j-1)^2 \tag{8}$$

Here, $\hat{y}$ is the output predicted by the nonlinear fuzzy model, Δu is increment control signal, w is the future set-point and λ is the control weighting sequence. The parameters N_2, N_u, N_1, called the prediction, control and minimum cost horizon respectively, where they define the intervals over which the optimization is carried out.

In the following, we represent MPC law by polynomial RST form. This representation allows to identify the RST controller parameters without using the classical computing (metaheuristic algorithms) of the predictive control.

The MPC controller is implemented under an RST form through the difference equation [28]:

$$\boldsymbol{S}(z^{-1})\Delta(z^{-1})u(t) = -\boldsymbol{R}(z^{-1})y(t) + \boldsymbol{T}(z^{-1})w(t) \tag{9}$$

with:

degree$[\boldsymbol{S}(z^{-1})] =$ degree$[B(z^{-1})]$,

degree$[\boldsymbol{R}(z^{-1})] =$ degree$[A(z^{-1})]$,
degree$[\boldsymbol{T}(z^{-1})] = N_2$.

In order to avoid the inversion of predictive matrices required by the FMPC, a new idea based on the coupling between the prediction algorithm and metaheuristic algorithms is used to overcome this problem.

4. **Particle Swarm Optimization-Based Predictive Control.**

The PSO is a population based swarm algorithm [20-23]. In the PSO computational algorithm, population dynamics simulates bio-inspired behavior, i.e., a "bird flock's" behavior where it involves sharing of the information between all members of the population, and allows particles to take profit from the discoveries and previous experience of all other particles during the search of food. In this work, this issue is used to determine the best parameters of RST controllers. The initial vector that contains the controller parameters is randomly chosen. The size of this vector consists of N_2, n_a and n_b, prediction horizon, degrees of $A(z^{-1})$ and $B(z^{-1})$, respectively.

Each particle in PSO has a randomized velocity, which allows it to move through the search space and keeps track of its coordinates in the solution space. Each column of the matrix is an RST regulator represented by a particle. The column parameters that give a better response from the closed loop system, obtained by minimizing of the cost prediction function, is used in the next iteration for the following calculus. This value is called *pBest* (personal best). Another best value that is tracked by the global version of the particle swarm optimizer is the overall best value (fitness). Its location, called *gBest* (global best), is obtained among all the particles in the population. The past best position of the particle itself and the best overall position in the entire swarm are employed to obtain new position for the particle in quest to minimize (or maximize) the fitness. In each time step, the PSO concept consists of changing the velocity of each particle flying towards its *pBest* and *gBest* location. The velocity is weighted by random terms, with separate random numbers being generated for velocities towards *pBest* and *gBest* locations respectively [23-26].

At each step n, by using the individual best position, *pBest*, and *gBest*, a new velocity for the ith particle is updated by

$$v_i(t+1) = w_r v_i(t) + c_1 r_1 (X_{pbest}(t) - x_i(t)) + c_2 r_2 (X_{gbest}(t) - x_i(t)) \tag{10}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{11}$$

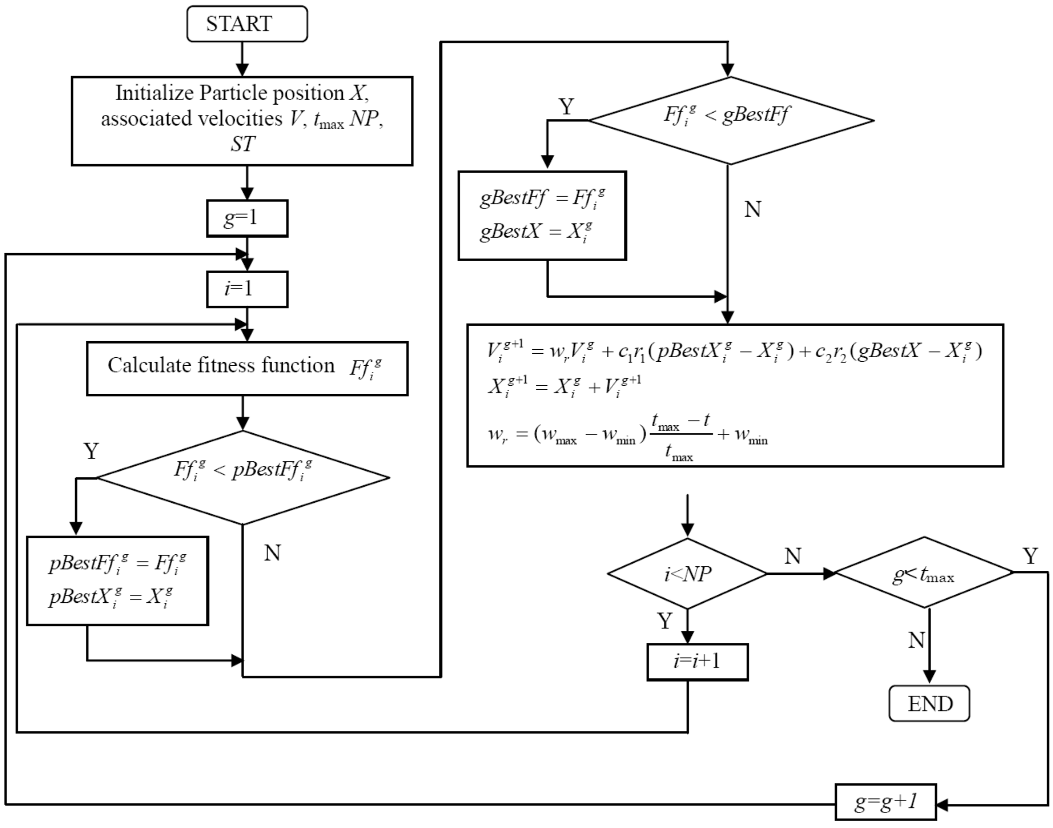
where each particle represents a potential solution and has a position represented by a position $x_i(t)$, r_1 and r_2 are two random values in the range $[0, 1]$, while c_1 and c_2 are the cognitive and social scaling parameters, respectively; w_r is inertia weight which controls the influence of previous velocity on the new velocity. The global search performance is good with large inertial weight while a small inertia weight facilitates a local search. The velocity $v_i(t)$ is limited within the range $[-v^{\max}, +v^{\max}]$. If the velocity violates these limits, it is forced to its proper values. The variable w_r is updated as

$$w_r = (w_{\max} - w_{\min})\frac{t_{\max} - t}{t_{\max}} + w_{\min} \tag{12}$$

where $w_{\max}$ and $w_{\min}$ denote the maximum and minimum of w_r respectively; $t_{\max}$ is a given number of maximum iterations.

The pseudo code of PSO is given in Figure 2, where *NP* and *ST* denote the number of particles in the population and the plant simulation time, respectively, $f(x_i(t))$ represents the objective function value of particle i at position x, while $f(x_{best}(t))$ represents the best function value in the population of solution *NP* at iteration count t.

```
    Initializing the velocity $v_i(t)$ and the position $x_i(t)$ of each particle
    **For** each particle $i$
            Calculate the fitness value $f(x_i(t))$,
        **If** the fitness value $f(x_i(t))$ is better than the
            Best fitness value $f(x_{pBest}(t))$ in history
        **Then**
                $x_{pBest}(t) = x_i(t)$
        **End**
        **If** $f(x_{pBest}(t)) < f(x_{gBest}(t))$ **then**
                $x_{gBest}(t) = x_{pBest}(t)$
        **End**
          Update the particle velocity $v_i(t+1)$ according to the velocity Equation (10),
          Update the particle position $x_i(t)$ according to the position Equation (11),
    **End**
While maximum iterations or minimum error criteria is reached.
```



START
Initialize Particle position X, associated velocities V, $t_{\max}$ NP, ST
g=1
i=1
Calculate fitness function Ff_i^g
$Ff_i^g < pBestFf_i^g$
Y
N
$pBestFf_i^g = Ff_i^g$
$pBestX_i^g = X_i^g$
$Ff_i^g < gBestFf$
Y
N
$gBestFf = Ff_i^g$
$gBestX = X_i^g$
$V_i^{g+1} = w_r V_i^g + c_1 r_1 (pBestX_i^g - X_i^g) + c_2 r_2 (gBestX - X_i^g)$
$X_i^{g+1} = X_i^g + V_i^{g+1}$
$w_r = (w_{\max} - w_{\min}) \frac{t_{\max} - t}{t_{\max}} + w_{\min}$
i<NP
Y
N
i=i+1
$g < t_{\max}$
Y
N
END
g=g+1

FIGURE 2. Flowchart representation of PSO algorithm. g: PSO iteration counter; i: particle counter.

As the aim is to minimize the error between the output of fuzzy model and reference model, in this paper, the mean cost function (MCF) in Equation (8) is used as a proper evaluation function. MCF is given by

$$MCF = \frac{1}{ST} \sum_{k=1}^{ST} J(k) \tag{13}$$

where ST is the plant simulation time, J is the cost function predictive and k is the iteration counter. Since the objective is to minimize cost function value, the fitness function is defined as follows:

$$Ff_i = \max(MCF) - MCF_i, \quad i = 1, \dots, NP \tag{14}$$

This function is evaluated by given the objective Function (8) of all the particles, and in each iteration we will select the best particle according to the objective function, that allows us to replace the weakest particles from the previous population with the strongest particles of the new population, and if the objective function value reaches the defined tolerance or the maximum iteration number is reached, so the algorithm is stopped (see Figure 3). The control strategy, of our approach can be represented by the following steps:

1. Initialize the structure and free parameters of the RLS
2. Fuzzify input variables
3. Calculate membership values of input
4. Calculate the Plant's dynamical behavior using 4th order Runge-Kutta Method
5. Calculate the fuzzy regressor vector
6. Stop condition if $\varepsilon >$ error
7. Else if update the parameters of systems by using RLS
8. Initialize the parameter of AFMPC
9. Calculate the matrix of prediction
10. Calculate the predicted output
11. Construct the cost function
12. Deduce the polynomial form of the fuzzy controller.
13. Initialize Particle position X, associated velocities V, $t_{\max}$, NP, ST
14. Calculate the cost function
15. $f(x_i(t)) = Ff_i$
16. Call the PSO algorithm
17. If $k < t_{\max}$
18. Go step 9
19. End

5. **Simulation Study.**

In this section, two highly nonlinear systems are selected to study the proposed approach. These systems are modeled by two FIs. The first example is to let the continuous-stirred tank reactor (CSTR). The second example is to let the surge tank. The values of the system parameters are realistic and used by several studies. PSO-based AFMPC is designed to realize closed-loop control for all these systems.

Example 5.1. *Control of a continuous-stirred tank reactor (CSTR)*

The benchmark plant represents a Continuous Stirred Tank Reactor where the model is presented in [33,34] and described by the following differentials equations:

$$\frac{dC_a(t)}{dt} = \frac{q}{v}\left(C_{a0} - C_a(t)\right) - k_0 C_a(t) e^{-\frac{E}{RT(t)}} \tag{15}$$

$$\frac{dT(t)}{dt} = \frac{q}{v}\left(T_0 - T(t)\right) + k_1 C_a(t) e^{-\frac{E}{RT(t)}} + k_2 q_c(t)\left(1 - e^{-\frac{k_3}{q_c(t)}}\right)\left(T_{c0} - T(t)\right) \tag{16}$$

The process describes the reaction that converts the product A into a new product B, the concentration $C_a(t)$ is the concentration of product A, $T(t)$ is the temperature of the mixture. The reaction is exothermic and it is controlled by a coolant flow whose rate is represented by $q_c(t)$. The temperature is controlled by changing the coolant flow and by controlling the temperature, and the concentration is also controlled. C_{a0} is the inlet

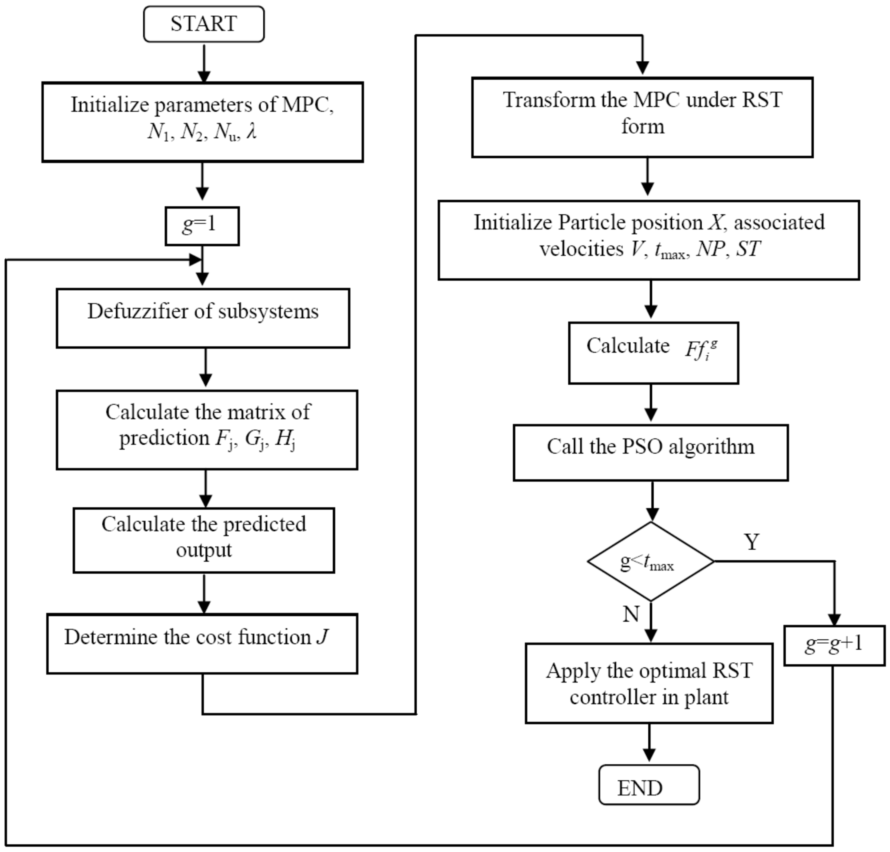

START
Initialize parameters of MPC, N_1, N_2, N_u, λ
g=1
Defuzzifier of subsystems
Calculate the matrix of prediction F_j, G_j, H_j
Calculate the predicted output
Determine the cost function J
Transform the MPC under RST form
Initialize Particle position X, associated velocities V, t_{max}, NP, ST
Calculate Ff_i^g
Call the PSO algorithm
g<t_{max}
Y
N
g=g+1
Apply the optimal RST controller in plant
END

Figure 3. Flowchart representation of PSO-based AFMPC algorithm. g: PSO iteration counter. Algorithm to solve the optimization problem of PSO-based AFMPC.

feed concentration, q is the process flow rate, T_0 and T_{c0} are the inlet feed and coolant temperatures. All these values are assumed constant at nominal values. In the same way, $k_0, E/R, v, k_1, k_2$ and k_3 are thermodynamic and chemical constants. The numerical values of these parameters are given in Table 1.

Table 1. CSTR model parameters

Parameter	Description	Nominal value
q	Process flow-rate	100 l/min
v	Reaction volume	100 l
k_0	Reaction rate constant	7.2×10^{10} min^{-1}
E/R	Activation energy	1×10^4 K
T_0	Feed temperature	350 K
T_{c0}	Inlet coolant temp.	350K
ΔH	Heat of reaction	2×10^5 cal/mol
C_p, C_{pc}	Specific heats	1 cal/g/K
ρ, ρ_c	Liquid densities	1×10^3 g/l
h_a	Heat transfer coeff.	7×10^5 cal/min/K
C_{a0}	Inlet feed concentration	1 mol/l

$$k_1 = \frac{\Delta H k_0}{\rho C_p} \quad k_2 = \frac{\rho_c C_{pc}}{\rho C_p v} \quad k_3 = \frac{h_a}{\rho_c C_{pc}}$$

The nominal conditions for a product concentration $C_a = 0.1$ mol/l are $T = 438.54$ K and $q_c = 103.41$ l/min.

Fuzzy modeling: the above nonlinear model is used to produce input-output time data. The sampling time is set to 0.083 min (5 s). The data is then used to develop a global fuzzy model as follows:

$$\begin{aligned} R^1: \ & \text{IF } q_c \text{ is } Q^1 \\ & \text{THEN } C_A^1(n+1) = a_{11}C_A(n-1) + \ldots + a_{1na}C_A(n-n_a) \\ & \qquad\qquad\qquad + b_{11}q_c(n-1) + \ldots + b_{1nb}q_c(n-n_b) \end{aligned} \tag{17}$$

$$\begin{aligned} R^2: \ & \text{IF } q_c \text{ is } Q^2 \\ & \text{THEN } C_A^2(n+1) = a_{21}C_A(n-1) + \ldots + a_{2na}C_A(n-n_a) \\ & \qquad\qquad\qquad + b_{21}q_c(n-1) + \ldots + b_{2nb}q_c(n-n_b) \end{aligned} \tag{18}$$

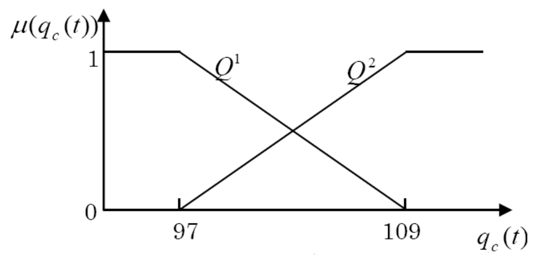


FIGURE 4. Definition of fuzzy sets Q^1 and Q^2 for FIs R^1 and R^2, respectively

The fuzzy model is structurally very simple and requires only two FIs, Figure 4, corresponding to the fuzzy sets Q_1 and Q_2. The open loop response with various step changes in the coolant flow rate shows that the identification of the fuzzy model can nearly perfectly describe the process dynamic behavior in Figure 5, and it also indicates that the process is highly nonlinear. The vector of parameters of ith rule is obtained by using the least squares method [35]:

$a_{11} = -1.7016$; $a_{12} = 0.7715$; $b_{11} = -0.2006$; $b_{12} = 0.2375$;

$a_{21} = -1.8480$; $a_{22} = 0.8984$; $b_{21} = -0.1397$; $b_{22} = 0.1973$.

The identified model can be observed in Figure 5; it can be observed that the quality of the model is very good and in fact the two signals appear overlapped; the error is plotted at the bottom of Figure 5, with another scale to make it visible.

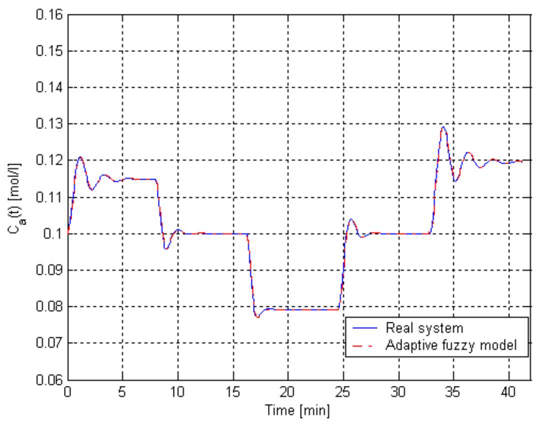
***i* – Switching system without disturbances**

The MPC parameters are selected according to the tuning rules given in Section 2. Since the minimum cost horizon $N_1 = 1$, the prediction horizon $N_2 = 10$, and the control horizon $N_u = 1$. The weight in the increment control $\lambda = 2 \times 10^{-3}$. The PSO algorithm parameters are chosen as follows:

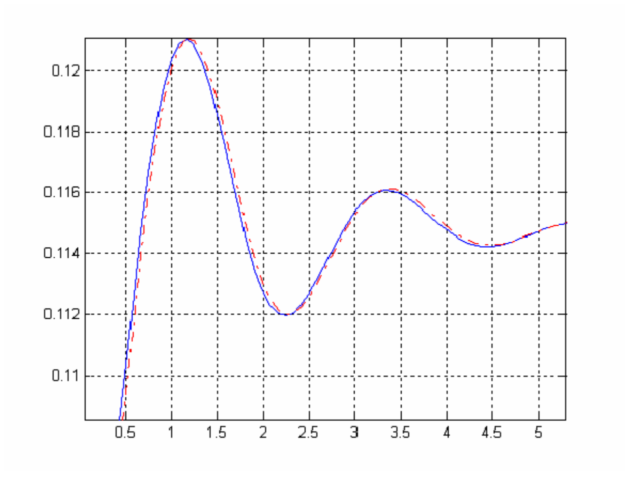
The maximum number of PSO iteration $t_{\max} = 150$, number of particles $NP = 10$, plant simulation time $ST = 50$ min, $r_1 = 0.2$, $r_2 = 0.4$, $w_{\min} = 0.35$, $w_{\max} = 0.85$, $c_1 = 1.5$ and $c_2 = 3.5$.

PSO algorithm can initialize the ranges of the research on the worst cases that will allow controlling the system in very critical situations. Its advantage lies in its ability to close to the best parameters while maintaining a good compromise between the desired performance and the various critical situations, so it can adapt to the change of the system parameters.

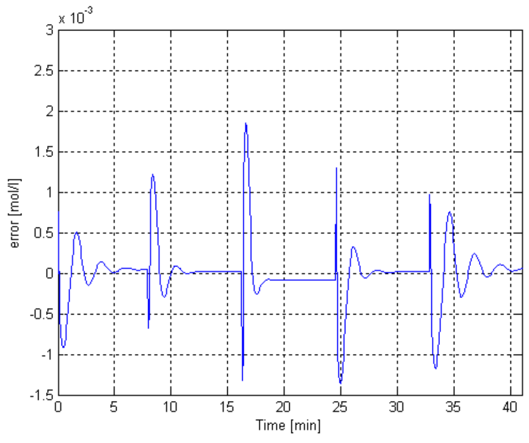
In PSO algorithm, the predictive cost function is calculated for each particle, and *pBest* and *gBest* are computed for every final time. A particle velocity Equation (10) is calculated for each particle and a particle position Equation (11) is updated. Considering the cost



(a)



(b)



(c)

Figure 5. (a) Validation of the fuzzy model, process output (solid line), model identified (dashed line), (b) zoom, (c) identification error

function of the prediction, PSO algorithm understands the behavior of the system and then at each iteration the particles enhance their performances to find the best controller parameters. This algorithm is run until maximum iteration or minimum cost criteria are obtained.

As mentioned in Section 4 for the size of the parameters vector to be optimized, we choose the degree of $A(z^{-1})$ and $B(z^{-1})$ respectively $n_a = 2$, $n_b = 2$, and $N_2 = 10$, so the degree of the RST polynomial is equal to 14, and also we have 14 parameters to search and optimize. The first three parameters represent the polynomial $\boldsymbol{R}$, the fourth parameter represents the polynomial $\boldsymbol{S}$ and the last ten parameters represent the polynomial $\boldsymbol{T}$. The initial values of particles are randomly generated in the first generation. The population size is set to 10 particles. Each particle represents a vector of optimized parameters, the vector dimension of position and velocity is 14×10.

The control quality the strategies PSO-based AFMPC controller is studied by a series of simulations. The response of the system with multi-step reference is given in Figure 6; the reference of C_a was changed from the initial point 0.14 mol/l to 0.12, to 0.11 and then to 0.09.

The dynamic response of the system is depicted in the same figure. Figure 6 shows details of the performance comparison between the PSO-based AFMPC and the AFMPC. It is important to remark the degradation of performance with the AFMPC controller when the concentration is closed to 0.14 mol/l or to 0.09 mol/l. Observe that the AFMPC controller is not able to stabilize the plant near this set points, so there is a risk if we want to reduce the concentration (see Figure 8). PSO is able to obtain good parameters so we can lower the concentration value down 0.09 mol/l and up 0.14mol/l, the command may well set the value of the concentration in any moment that gives us a good security for the reactor.

Mention may also be based on the work published in the book [6] that the command with the PID control deteriorates when working in a concentration 0.12 mol/l. We see that the PID control is not able to stabilize the system around this point.

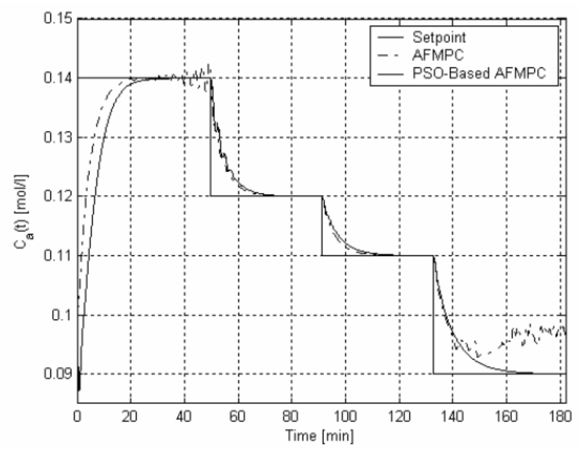


FIGURE 6. Comparison of the closed-loop dynamic responses by the PSO-based AFMPC (dash line) and an AFMPC (solid line)

***ii* – Switching system with disturbances**

In a second test, the disturbances on the system output in different times have been applied to validate the tracking of the reactor concentration. Thus, a disturbance of 0.002 mol/l at time t = 66.4 min and t = 107.9 min is added. Figure 9, illustrates the disturbance rejection performance of the PSO-based AFMPC controller. The results show that the adaptive controller has the ability to keep the process stable and regulate the outlet concentration at its desired set-point value. These features prove the good performance of the control law, even in the presence of the step changes of the disturbances and of the set-point value (see Figure 10).

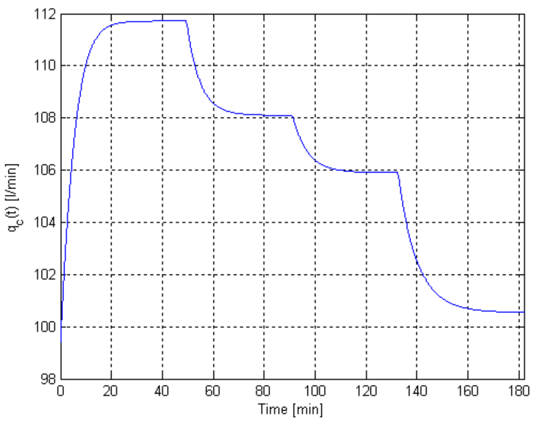


FIGURE 7. Control performance of the PSO-based AFMPC controller

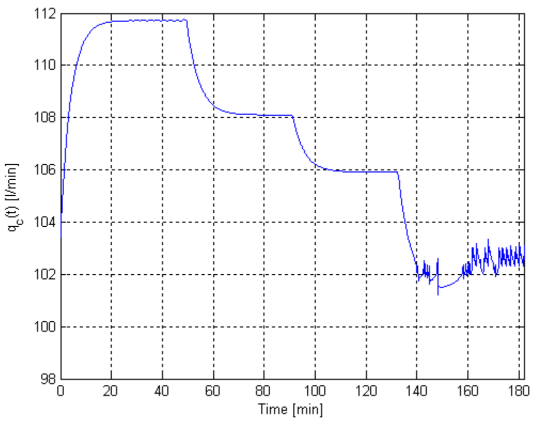


FIGURE 8. Control performance of the AFMPC controller

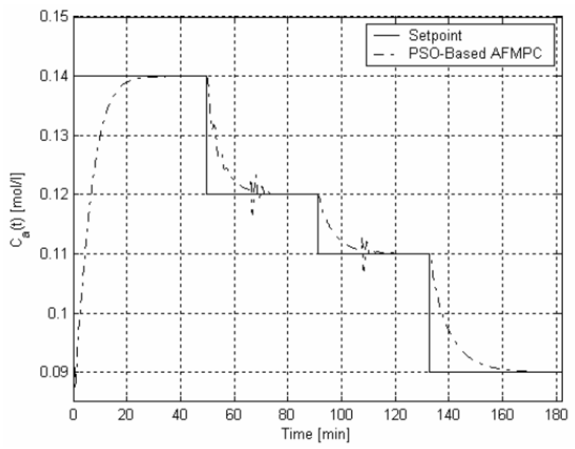


FIGURE 9. Response of the CSTR using a PSO-based AFMPC with external disturbances

Example 5.2. *Control of surge tank model*

The behavior of the surge tank system, shown in Figure 11, it is fed by a pump driven by a current $i(t)$.

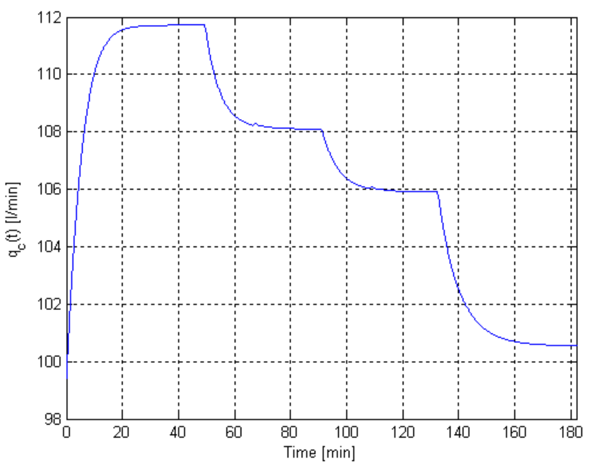


FIGURE 10. Control performance of the PSO-based AFMPC with external disturbances

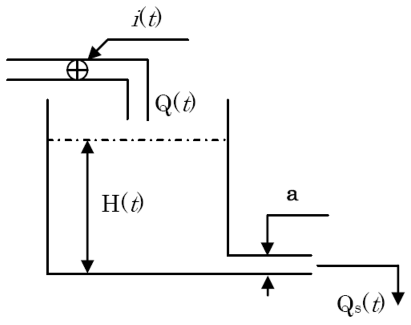


FIGURE 11. The surge tank system

In Figure 11, $Q(t)$ is the feed rate, $i(t)$ is the supply current of the pump, $H(t)$ is the liquid level in the tank, $Q_s(t)$ is the output Flow, a is the section of the output channel, A is the section of the tank and H_s is the water level in the output channel. This system can be represented by the following differential equations:

Model of the valve:

$$\frac{dQ(t)}{dt} + k_0 Q(t) = k_1 i(t) \tag{19}$$

The change in water level in the tank is given by:

$$\frac{dV(t)}{dt} = A\frac{dH(t)}{dt} = Q(t) - Q_s(t) \tag{20}$$

where $Q_s(t) = 0.6a\sqrt{2g\,(H(t) - H_s)}$.

TABLE 2. Specification of the surge tank

Parameter	Description	Normal operation condition
H_0	Initial value of tank level	0.15 m
H_s	Initial value of the output channel level	0.015 m
a	Section of the channel output	0.0001 m^2
A	Section of the tank	0.04 m^2
q_0	the initial flow	0.0001 m^3/s
K_0	Constant	1
K_1	Constant	0.1

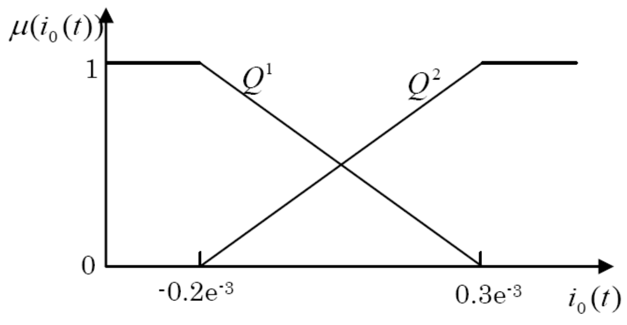


FIGURE 12. Definition of fuzzy sets Q^1 and Q^2 for FIs R^1 and R^2, respectively

Fuzzy modeling: The fuzzy model was identified using data from the real process, sampled with the period $T_s = 0.1$ s. For a good approximation of the plant, we suppose that the subsystems are in the third order. The model consists of two rules of the form

$$\begin{aligned} R^1 &: \text{IF } i_0 \text{ is } Q^1 \\ &\quad \text{THEN } H^1(k+1) = a_{11}H(k-1) + \ldots + a_{1na}H(k-n_a) \\ &\qquad\qquad + b_{11}i_0(k-1) + \ldots + b_{1nb}i_0(k-n_b) \end{aligned} \tag{21}$$

$$\begin{aligned} R^2 &: \text{IF } i_0 \text{ is } Q^2 \\ &\quad \text{THEN } H^2(k+1) = a_{21}H(k-1) + \ldots + a_{2na}H(k-n_a) \\ &\qquad\qquad + b_{21}i_0(k-1) + \ldots + b_{2nb}i_0(k-n_b) \end{aligned} \tag{22}$$

The fuzzy model is structurally very simple and requires only two FIs, Figure 12 corresponds to the fuzzy sets Q_1 and Q_2. The vector of parameters of ith rule is obtained by using the recursive least squares (RLS) [35]. This fuzzy model is used to represent the process model in the controller:

$a_{11} = 1.941$; $a_{21} = -0.0409$; $a_{31} = -0.0526$; $b_{11} = -0.1129$; $b_{21} = -0.1107$;
$a_{12} = 0.8419$; $a_{22} = 0.0810$; $a_{32} = 0.0763$; $b_{12} = 0.1667$; $b_{22} = 0.1682$.
The predictive controller was implemented using the following parameters:
Prediction horizon $N_2 = 10$, control horizon $N_u = 1$, minimum horizon $N_1 = 1$, the weight in the increment control $\lambda = 40$. We choose the degree of $A(z^{-1})$ and $B(z^{-1})$ respectively $n_a = 3$, $n_b = 2$, and $N_2 = 10$, so the degree of the RST polynomial is equal to 15, and consequently we have 15 parameters to search and optimize.

The first four parameters represent the polynomial $\boldsymbol{R}$, and the second fifth parameter represents the polynomial $\boldsymbol{S}$. Finally, the last ten parameters represent the polynomial $\boldsymbol{T}$.

The PI-PSO algorithm uses the objective function, which is the Integral of the absolute value of Error (IAE), whose values are minimized. In order to show the differences between the PSO-based AFMPC controller and PI-PSO controller, a comparative study will be necessary to show the differences in their performances.

$$IAE = \int |e(t)|dt \tag{23}$$

In the case of PI-PSO, the parameters of the algorithm are chosen as follows:

The maximum number of PSO iteration $t_{\max} = 200$, number of particles $NP = 10$, plant simulation time $ST = 60$s, $r_1 = \text{rand}(dim, NP)$, $r_2 = \text{rand}(dim, NP)$, $w_r = 0.1$, $c_1 = 0.5$ and $c_2 = 2.5$ where dim is the problem dimension ($dim = 2$).

In the case PSO-based AFMPC the parameters of the algorithm are chosen as follows:

The maximum number of PSO iteration $t_{\max} = 200$, number of particles $NP = 10$, plant simulation time $ST = 60$s, $r_1 = \text{rand}(dim, NP)$, $r_2 = \text{rand}(dim, NP)$, $w_r = 1e^{-3}$, $c_1 = 0.5$ and $c_2 = 4.5$, where $dim = (N_2 + n_a + n_b) = 15$.

To evaluate performance of each controller, we consider the whole characteristic of the each controller, such as maximum overshoot, rise time, settling time and a steady-state error, to have an accurate comparison.

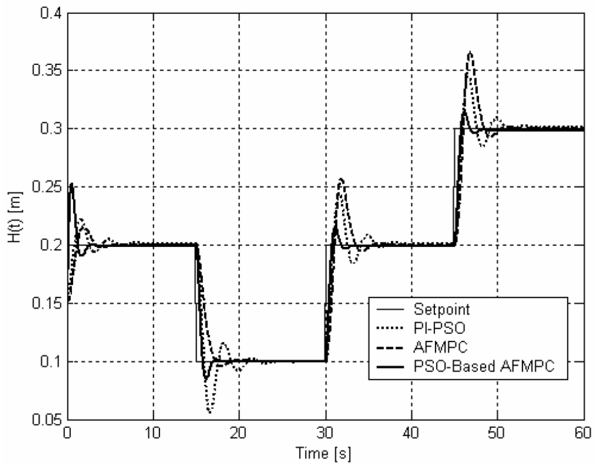


FIGURE 13. Comparison of the closed-loop dynamic responses by the PSO-based AFMPC, an PI-PSO and an AFMPC

TABLE 3. Comparison of performance of the PI-PSO, AFMPC and PSO-based AFMPC

	PI-PSO				AFMPC				PSO-based AFMPC			
Time (s)	0-15	15-30	30-45	45-60	0-15	15-30	30-45	45-60	0-15	15-30	30-45	45-60
Maximum overshoot (%)	12.5	50	25.75	17.33	8	4.5	28.25	21.66	26.3	15	6.2	5
Rise time (s)	0.7	0.74	0.74	0.748	1.24	2.2	1.09	1.055	0.2	0.8	0.82	0.81
Settling time (s)	4	2.05	5.6	4.48	3	3.72	4.75	4.45	2.04	1.85	1.69	1.6
Steady state error	0.002	0.0	0.002	0.003	0.0	0.0	0.0	0.0	0.0002	0.0	0.0	0.0

This table shows the performances obtained these methods. In each interval time, we have changed the set point for evaluating each method to control a highly nonlinear system. The PSO-based AFMPC method can generate a high quality solution within shorter calculation time and it tends to converge very fast compared to other methods.

The comparison shows some interesting results. The AFMPC is not capable of keeping the level at the desired value. It is important to observe that with PSO-based AFMPC the settling time has been reduced 3 times comparing with that obtained from the AFMPC without any increase in overshoot. Between 45-60 s we can see from the overshot, rise time and settling time that obtained by the PSO-PI or AFMPC controller are degraded. With PSO-based AFMPC the overshoot is reduced more than 3 times compared with that obtained from PI-PSO and more than 4 times compared with that obtained from the AFMPC controller. The same observation can be made for the settling time, where in the PSO-based AFMPC case we notice a reduction of nearly 3 times compared with that obtained from PI-PSO and more than 2 times with AFMPC. So, the PSO-based AFMPC is able to keep better stability with less control effort applied. As a conclusion, a large overshoot in the transient response with PI-PSO and AFMPC can damage the electro-valve of the tank.

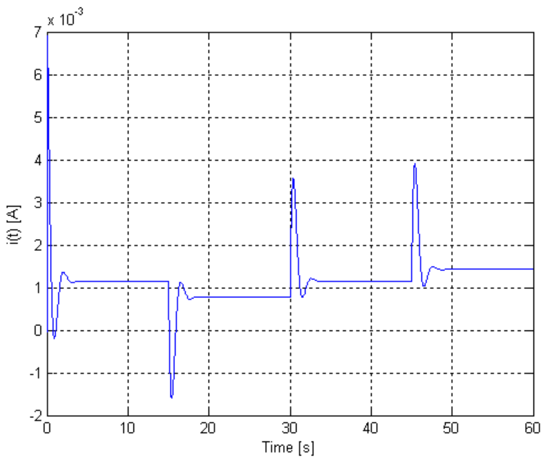


FIGURE 14. Control performance of the PSO-based AFMPC controller

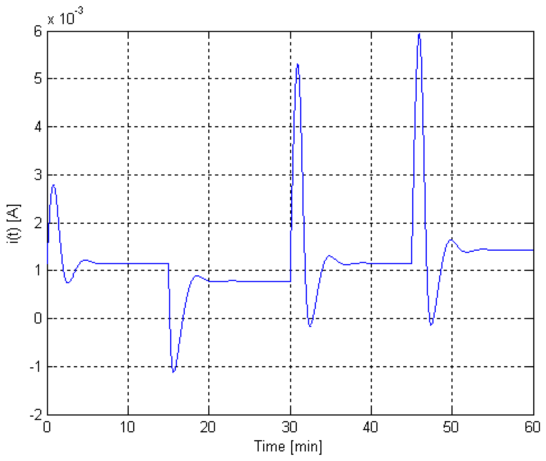


FIGURE 15. Control performance of the AFMPC controller

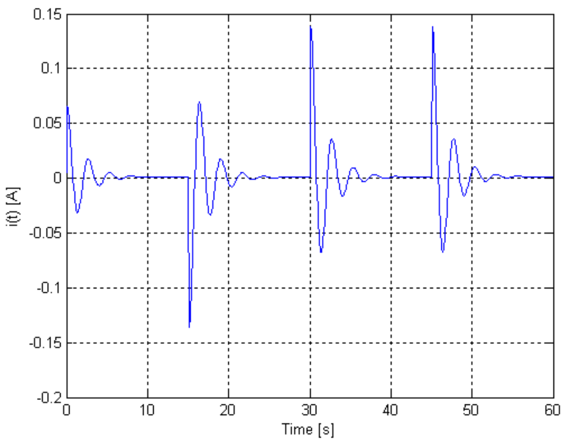


FIGURE 16. Control performance of the PI-PSO controller

6. **Conclusion.** In this paper, we have introduced the PSO-based AFMPC and PI controller applied to highly nonlinear systems. An optimization approach of polynomial RST fuzzy predictive control is developed using PSO algorithm. The proposed approach is based on the advantage of the Takagi-Sugeno fuzzy system and the metaheuristic optimization PSO algorithm in a new structure RST controller. The advantage of this structure is their ability to handle highly nonlinear systems and keep a good stability in terms of overshoot, rise time and settling time including disturbances. Compared with other similar existing methods, the PSO-based AFMPC algorithm enhances the convergence and accuracy of the controller optimization, which is much easier for implementation in real time. Indeed, it helps to avoid the computing complexity of the AFMPC control law. Future work consists of the constraints consideration in the AFMPC problem with other versions of PSO and ACO algorithm, applied to a real process.

Acknowledgment. The authors would like to thank associate editor and reviewers for valuable comments and suggestions for improving the presentation of this paper. They also thank Prof. L. Guyot (UPJV) for reviewing the language.

REFERENCES

[1] D. W. Clarke, C. Mohtadi and P. S. Tuffs, Generalized predictive control, Part I: The basic algorithm, Part II: Extensions and interpretation, *Automatica*, vol.23, no.2, pp.137-160, 1987.

[2] Y. Zhang, T. Chai, H. Wang, J. Fu, L. Zhang and Y. Wang, An adaptive generalized predictive control method for nonlinear systems based on ANFIS and multiple models, *IEEE Transactions on Fuzzy Systems*, vol.18, no.6, pp.1070-1082, 2010.

[3] S. A. Larrinaga, M. A. R. Vidal, E. Oyarbide and J. R. T. Apraiz, Predictive control strategy for DC/AC converters based on direct power control, *IEEE Transactions on Industrial Electronic*, vol.54, no.3, pp.1261-1271, 2007.

[4] J. Licheng, R. Kumar and N. Elia, Model predictive control-based real-time power system protection schemes, *IEEE Transactions on Power Systems*, vol.25, no.2, pp.988-998, 2010.

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

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

[7] Y. J. Liu, W. Wang, S. C. Tong and Y. S. Liu, Robust adaptive tracking control for nonlinear systems based on bounds of fuzzy approximation parameters, *IEEE Transactions on Systems, Man, Cybernetics, Part A: Systems and Humans*, vol.40, no.1, pp.170-184, 2010.

[8] M. Chadli, On the stability analysis of uncertain fuzzy models, *International Journal of Fuzzy Systems*, vol.8, no.4, pp.224-231, 2006.

[9] M. Chadli and T.-M. Guerra, LMI solution for robust static output feedback control of Takagi-Sugeno fuzzy models, *IEEE Transactions on Fuzzy Systems*, vol.20, no.6, 2012.

[10] P. P. Angelov and D. P. Filev, An approach to online identification of Takagi-Sugeno fuzzy models, *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, vol.34, no.1, pp.484-498, 2004.

[11] H. N. Nounou and K. M. Passino, Fuzzy model predictive control: Techniques, stability issues, and examples, *Proc. of IEEE International Symposium Intelligent Control Intelligent Systems and Semiotics*, pp.423-428, 1999.

[12] A. Flores, D. Saez, J. Araya, M. Berenguel and A. Cipriano, Fuzzy predictive control of a solar power plant, *IEEE Transactions on Fuzzy Systems*, vol.13, no.1, pp.58-68, 2005.

[13] S. Bououden, O. Benelmir, S. Ziani and S. Filali, A new adaptive fuzzy model and output terminal constraints in predictive control, *International Journal of Information and Systems Sciences*, vol.3, no.1, pp.25-35, 2007.

[14] 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.

[15] K. Belarbi and F. Megri, A stable model-based fuzzy predictive control based on fuzzy dynamic programming, *IEEE Transactions on Fuzzy Systems*, vol.15, no.4, pp.746-754, 2007.

[16] H.-M. Lee, R. Jea and J.-S. Su, Fuzzy nonlinear programming for inventory based on confidence interval *ICIC Express Letters*, vol.6, no.5, pp.1387-1392, 2012.
[17] W. Ma, J. Song and W. Wang, Dynamic filled algorithm for global optimization of nonlinear programming, *International Conference on Information Networking and Automation*, pp.V1-23-V1-27, 2010.
[18] T. Niknam, H. D. Mojarrad and M. Nayeripour, A new hybrid fuzzy adaptive particle swarm optimization for non-convex economic dispatch, *International Journal of Innovative Computing, Information and Control*, vol.7, no.1, pp.189-202, 2011.
[19] 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.
[20] X. S. Yang, *An Introduction with Metaheuristic Application*, John Wiley & Sons, 2010.
[21] A. Núñez, D. Sáez, S. Oblak and I. Škrjanc, Fuzzy-model-based hybrid predictive control, *Isa Transactions*, vol.48, no.1, pp.24-31, 2009.
[22] M. Rashid and A. R. Baig, PSOGP: A genetic programming based adaptable evolutionary hybrid particle swarm optimization, *International Journal of Innovative Computing, Information and Control*, vol.6, no.1, pp.287-296, 2010.
[23] Y. Shi and R. C. Eberhart, Parameter selection in particle swarm optimization, *Evolutionary Programming VII, LNCS*, vol.1447, pp.591-600, 1998.
[24] C. M. Lee and C. N. Ko, Time series prediction using RBF neural networks with a nonlinear time-varying evolution PSO algorithm, *Neurocomputing*, vol.73, pp.449-460, 2009.
[25] K. D. Sharma, A. Chatterjee and A. Rakshit, A Hybrid approach for design of stable adaptive fuzzy controllers employing Lyapunov theory and particle swarm optimization, *IEEE Transactions on Fuzzy Systems*, vol.17, no.2, pp.329-342, 2009.
[26] Z. Bingül and O. Karahan, A fuzzy logic controller tuned with PSO for 2 DOF robot trajectory control, *Expert Systems with Applications*, vol.38, pp.1017-1031, 2011.
[27] J. Yu, S. Wang and L. Xi, Evolving artificial neural networks using an improved PSO and DPSO, *Neurocomputing*, vol.71, pp.1054-1060, 2008.
[28] P. Boucher and D. Dumur, *La Commande Prédictive*, Editions Technip, Collection Méthodes et Pratiques de l'Ingénieur, Paris, 1996.
[29] S. Bououden, S. Filali and K. Kemih, Adaptive fuzzy tracking control for unknown nonlinear systems, *International Journal of Innovative Computing, Information and Control*, vol.6, no.2, pp.541-549, 2010.
[30] C.-P. Huang, Model based fuzzy control with affine T-S delayed models applied to nonlinear systems, *International Journal of Innovative Computing, Information and Control*, vol.8, no.5(A), pp.2979-2993, 2012.
[31] L. J. Chenand and K. S. Narendra, Identification and control of a nonlinear discrete-time system based on its linearization: A unified framework, *IEEE Transactions on Neural Networks*, vol.15, no.3, pp.663-673, 2004.
[32] D. W. Clarke and C. Mohtadi, Properties of generalized predictive control, *The 10th World Congress IFAC*, pp.63-74, 1987.
[33] M. Henson and D. Seborg, Input-output linearization of general nonlinear processes, *AIChE Journal*, vol.36, no.11, pp.1753-1757, 1990.
[34] G. Lightbody and G. Irwin, Nonlinear control structures based on embedded neural system models, *IEEE Transactions on Neural Networks*, vol.8, pp.553-567, 1997.
[35] K. Astrom and B. Wittenmark, *Adaptive Control*, Addison-Wesley Editions, 1989.

Optimal design of TS fuzzy controller for a nonlinear system using a new adaptive particle swarm optimization algorithm.Â D. A. Linkens, M.Y. Chen, Input selection and partition validation for fuzzy modeling using neural network, Fuzzy Sets and Systems, Vol.107, pp. 299308, 1999. Z. Fang, N. Song, L. Wang, Design and Implementation of a Novel Fuzzy Controller with DSP for Rotary Inverted Pendulum, IEEE Int. Conf. on Control and Decision, pp.61226127, 2009. M. Setnes, Supervised fuzzy clustering for rule extraction, IEEE Trans. fuzzy systems, Vol. 8, No. 4, pp. 416424, Aug. In computational science, particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known