Chapter 5

PID Control for Takagi-Sugeno Fuzzy Model

Taieb Adel and Chaari Abdelkader

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.74295

Abstract

In this chapter, we deal with the problem of controlling Takagi-Sugeno (TS) fuzzy model by PID controllers using the particle swarm optimization (PSO). Therefore, a new algorithm is proposed. This algorithm relies on the use of a new objective function taking into account both the performance indices and the error signal. The advantages of this approach are discussed through simulations on a numerical example.

Keywords: nonlinear system, TS fuzzy model, PID, self-tuning of PID controller, PSO

1. T-S fuzzy model

The theory of modeling based on multiple models has evolved greatly. Indeed, several techniques have been developed in the literature. The common feature of all these modeling techniques is the decomposition of the dynamic behavior of the system into a number of operating zones. Each zone is characterized by a local linear model. Fuzzy logic based on the use of linguistic rules, heuristic strategies and the operator's know-how. Subsequently, it has undergone a major evolution mainly in Japan where it has been applied in several industrial applications. This type of model proposed by Takagi and Sugeno (1985) makes it possible to express a nonlinear system in several locally linear subsystems. The validity of each local model is defined by a weighting function with bounded support.

The TS model is built on a set of rules of type:

- R^i: "IF premises THEN consequence"

where the premises are obtained from the linguistic propositions allowing the evaluation of the weighting functions and where the consequences corresponding to the local models.

We consider a class of nonlinear systems defined by:

$$y(t+1) = f(X(t)) \tag{1}$$

with the regressor vector $X(t)$ is:

$$X(t) = [y(t), y(t-1), \ldots, y(t-m, u(t), u(t-1), \ldots, u(t-m)] \tag{2}$$

where k represents the discrete time, n and m denote, respectively, the number of delayed output and the number of delayed input. The function $f(x(t))$ is approximated by a TS fuzzy model which is charities by consequent rules that are local linear function of the input variables [1]. The fuzzy rules of the TS model take the following general form:

$$R^i : \text{if } X_1 \text{ is } A_1^i \text{ and if } X_z \text{ is } A_z^i \text{ Then } y_i(t) = [X(t)\ 1]\theta_i^T \tag{3}$$

where R^i denotes the ith IF-THEN rule, r is the number of rules, $A_j^i\ (j = 1, \ldots, z)$ is the fuzzy subset, $u(t)$ is the system input variable, $y(t)$ is the system output, $\theta_i = [a_{i1}, a_{i2}, \ldots, a_{iz}, b_{i0}]$ is the parameter vector of the corresponding local linear model. Let $\mu_i(X(t))$ is the normalized membership function of the inferred fuzzy set A^i, where $A^i = \Pi_{j=1}^z A_j^i$. The final output is calculated as the average of the outputs corresponding to the rules R^i, weighted by the normalized degree of completion (membership), according to the following expression:

$$\widehat{y} = \sum_{i=1}^{r} \mu_{it} y_i \tag{4}$$

The membership values μ_{it} have to satisfy the following conditions:

$$\mu_{it} \in [0\ 1] \quad i = 1, \ldots, r \tag{5}$$

$$\sum_{i=1}^{r} \mu_{it} = 1 \quad t = 1, \ldots, N \tag{6}$$

$$0 < \sum_{k=1}^{N} \mu_{it} < N \quad i = 1, \ldots, r \tag{7}$$

Once the parameters of the premises are fixed, the parameters of the consequent for each rule can be obtained using the recursive weighted least squares technique, using the values of the membership degrees of the fuzzy partition matrix of the classification process as weights [2]:

The steps of the WRLS method are summarized in the following algorithm:

Initialize: $\boldsymbol{\theta}_{ig(0)} = 0$ and $\mathbf{P}_{i(0)} = \alpha_i I$.

for $g = 1, \ \ldots, \ c_i$

http://dx.doi.org/10.5772/intechopen.74295

$$\boldsymbol{\theta}_{ig(t)} = \boldsymbol{\theta}_{ig(t-1)} + \mathbf{L}_{i(t)}\left(y_{i(t)} - \left[\mathbf{x}_{i(t)}\ 1\right]\boldsymbol{\theta}_{ig(t-1)}^{T}\right) \tag{8}$$

$$\mathbf{L}_{i(t)} = \frac{\mathbf{P}_{i(t-1)}\left[\mathbf{x}_{i(t)}\ 1\right]^{T}}{1/\mu_{igt} + \left[\mathbf{x}_{i(t)}\ 1\right]\mathbf{P}_{i(t-1)}\left[\mathbf{x}_{i(t)}\ 1\right]^{T}} \tag{9}$$

$$\mathbf{P}_{i(t)} = \mathbf{P}_{i(t-1)} - \mathbf{L}_{i(t)}\left[\mathbf{x}_{i(t)}\ 1\right]\mathbf{P}_{i(t-1)} \tag{10}$$

with $k = 1, \ldots, N$, $\mathbf{P}_{i(t-1)} \in \mathfrak{R}^{(M_i+1)\times(M_i+1)}$ and $\mathbf{L}_{i(t)} \in \mathfrak{R}^{(M_i+1)}$.

end for

2. PID control by pole placement

This section is intended to model the digital PID controller in a new form RST. The control structure RST is the establishment of three polynomials $R(q^{-1})$, $S(q^{-1})$ and $T(q^{-1})$. According to the first section, the local linear systems can be represented by:

$$Y(q^{-1})A(q^{-1}) = B(q^{-1})u(q^{-1}) \tag{11}$$

Indeed, the vector x_k takes the following form: $x_k = [-y(k-1) - y(k-2)\ldots - y(k-n)\,u(k-1)\ldots u(k-m)]$. This form is heard when determining PID controller parameters by the method of poles placement. **Figure 1** shows the standard form of RST controller.

In this chapter, we consider only two branches R and S, as shown in **Figure 2**, that is to say:

$$T(q^{-1}) = R(q^{-1}) \tag{12}$$

The transfer function in a closed loop is given by:

$$H_{bf} = \frac{B(q^{-1})}{A(q^{-1})S(q^{-1}) + B(q^{-1})R(q^{-1})} \tag{13}$$

again:

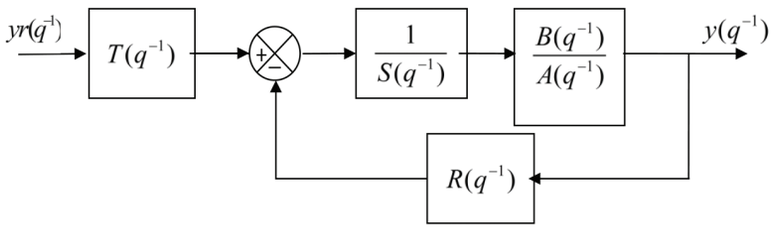

Figure 1. Standard form of RST.

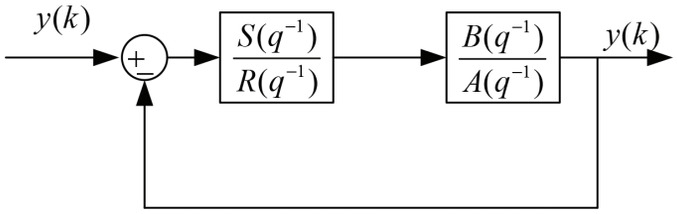

Figure 2. RST structure.

$$H_{bf} = \frac{B(q^{-1})}{P'(q^{-1})} \tag{14}$$

with:

$$\begin{aligned} &e(k) = yr(k) - y(k) \\ &B(q^{-1}) = b_1 q^{-1} + b_2 q^{-2} \\ &A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} \\ &R(q^{-1}) = r_0 + r_1(q^{-1}) + r_2(q^{-2}) \\ &S(q^{-1}) = (1 - q^{-1})(1 + s_1(q^{-1})) \end{aligned} \tag{15}$$

For a characteristic polynomial $P^1(q^{-1})$, the poles of the closed loop transfer function are imposed to arrive at the performances required by the follow-up of the specifications.

This problem boils down to solving the following equation:

$$P'(q^{-1}) = A(q^{-1})S(q^{-1}) + B(q^{-1})R(q^{-1}) \tag{16}$$

From the after Scheme 4, the equation of regulator is written by:

$$C(q^{-1}) = K_P\left(1 + \frac{1}{T_I(1 - q^{-1})} + \frac{\frac{N_f T_d}{T_d + N_f Te}(1 - q^{-1})}{1 - \frac{T_d}{T_d + N_f Te}q^{-1}}\right) \tag{17}$$

To simplify Eq. (17), we substitute:

$$a = \frac{Te}{T_I} \tag{18}$$

$$b = \frac{T_d}{T_d + N_f Te} \tag{19}$$

from which we obtain the following standard form:

$$C(q^{-1}) = K_p\left(1 + \frac{a}{1 - q^{-1}} + N_f b\frac{1 - q^{-1}}{1 - bq^{-1}}\right) \tag{20}$$

http://dx.doi.org/10.5772/intechopen.74295

The parameters of the digital PID regulator (r_0, r_1, r_2 and s_1) are chosen according to the desired poles defined by the polynomial $P(q^{-1})$.

The characteristic polynomial can be in the following form:

$$P'(q^{-1}) = 1 + p_1 q^{-1} + p_2 q^{-2} \tag{21}$$

The values of p_1 and p_2 are chosen from the specifications imposed by the specifications (rise time, damping, overshoot, etc.) defined in general by comparing the behavior of the process to that of continuous system of second order.

$$\begin{cases} p_1 = -2e^{\zeta} w_n Te cos\left(w_n Te\sqrt{1-\zeta^2}\right) \\ p_1 = e^{-2\zeta w_n Te} \end{cases} \tag{22}$$

where *zeta* is a damping coefficient and w_n is a pulsation. We wrote:

$$P(q^{-1}) = A(q^{-1})S(q^{-1}) + B(q^{-1})R(q^{-1}) \tag{23}$$

By identification, we find:

$$\begin{cases} b_1 r_0 + s_1 = p_1 + 1 - a_1 \\ b_2 r_0 + b_1 r_1 + (a_1 - 1)s_1 = p_2 + a_1 - a_2 \\ b_2 r_1 + b_1 r_2 + (a_2 - a_1)s_1 = a_2 \\ b_2 r_2 - a_2 s_1 = 0 \end{cases} \tag{24}$$

The control law is:

$$u(q^{-1}) = \frac{R(q^{-1})}{S(q^{-1})} e(q^{-1}) \tag{25}$$

From where:

$$u(k) = r_0 e(k) + r_1 e(k-1) + r_2 e(k-2) + (1 - s_1)u(k-1) + s_1 u(k-2) \tag{26}$$

The structure of the corrector is the standard structure discretized by the approximation upper rectangles, and to find the values of the PID parameters will take the following expression:

$$\begin{cases} K_p = \dfrac{r_0 s_1 - r_1 - (2 + s_1)r_2}{(1 + s_1)^2} \\ T_i = \dfrac{TeK_p(1 + s_1)}{r_0 + r_1 + r_2} \\ T_d = \dfrac{Te\left(s_1^2 r_0 - s_1 r_1 + r_2\right)}{K_p(1 + s_1)^3} \end{cases} \tag{27}$$

3. PID controller based on PSO

3.1. PSO algorithm

Particle swarm optimization (PSO) is a stochastic technique based on collective intelligence, inspired by nature. It was developed by Kennedy and Eberhart [3]. The PSO algorithm is inspired by collective behavior in certain social animals such as fish and migratory birds. This algorithm shares many similarities with evolutionary computational techniques such as genetic algorithms. Indeed, the latter are initialized with random solutions and search for optimums by updating generations involved. However, the PSO has no evolutionary operator such as the crossing and the mutation in the image of genetic algorithms.

In the PSO, each individual of the population is called particle, while the population is known as swarm. It should be noted that a particle can benefit from the movements of other particles in the same population to adjust its position and velocity during the optimization process. Each individual uses local information to which he can access the movement of his nearest neighbors to decide his own move. Very simple rules like "staying close to other people," "going in the same direction" and "going at the same speed" are enough to maintain the cohesion of the whole group. At each displacement, the performance of each particle is measured by its position and velocity by minimizing a performance function called fitness [4].

The PSO's basic algorithm works on a population called a swarm of possible solutions, which are called particles. These particles are placed randomly in the search space of the objective function. At each iteration, the particles move, taking into account their best position (selfish displacement) but also the best position of its vicinity. In fact, the new speed is calculated from the following formula [3]:

$$V_{id}(t+1) = wV_{id}(t) + c_1 r_1 \left(p_{id}(t) - x_{id}(t)\right) + c_2 r_2 \left(p_{gd}(t) - x_{id}(t)\right) \tag{28}$$

In this equality, t is the number of iteration, $V_{id}(t)$ and $x_{id}(t)$ stand for separately the speed of the particle i at its t times and the d-dimension quantity of its position, c_1 and c_2 are the acceleration coefficients, r_1 and r_2 are two random numbers drawn uniformly in $[01]$. p_{id} and p_{gd} are, respectively, the best position reached and the best position of the vicinity reached of the particle i and w is an inertial coefficient defined by:

$$w = w_{\max}\left(\frac{w_{\max} - w_{\min}}{iter_{\max}}\right) iter \tag{29}$$

where $iter_{max}$ is the maximum of iteration in evolution process, $w_{\min}$ and $w_{\max}$ are the minimum and maximum values of w, respectively, and $iter$ is the current value of iteration. The position of the particle can then be determined by the speed that we have just calculated:

$$x_{id} = x_{id} + V_{id} \tag{30}$$

We generate $V_{id}(0)$ and $x_{id}(0)$ at the beginning of our algorithm. The PSO algorithm stops if one of these convergence criteria is reached:

http://dx.doi.org/10.5772/intechopen.74295

- the maximum number of iterations t_{max} is reached;
- the speed variation tends to zero; and
- the fitness function is satisfied.

3.2. Tuning of PID using PSO optimization

The PSO optimization module complete the self-tuning of PID parameters with a microprocessor that achieves the optimum of PID parameters. These parameters are used to retune the PID controller in PID controller module. To seek the optimum parameters *kp*, *ki* and *kd* of PID controller, PSO program should search in D-dimensional search space. The function optimization problem can be viewed as a 3-dimensional space in this chapter. That is, tuning of PID controller parameters is to search optimization value in *kp*, *ki* and *kd*, the 3-dimensionsal search spaces. With the optimized parameters based on PSO algorithm, the PID controller can achieve the optimal properties, that is, a fast system with a minimum of overrun, there is a compromise between performance and minimum energy [5].

In most cases, PID controller work with an error signal (e) that is calculated from the process variable (y) and setpoint (y_r). The error represents the deviation of the process variable from the setpoint. Then, the error signal is described as:

$$e(t) = y_r(t) - y(t) \tag{31}$$

A PID controller optimized with PSO algorithm was developed for a TS Fuzzy system. It was also called the PSO-PID controller. PSO algorithm is mainly utilized to determine three optimal PID gains.

Adjusting the parameters of a PID controller can be considered as an optimization problem where it is a matter of finding the optimal solution of the gains of the controller in a predefined search space in order to allow the system to have certain desired performances. In this context, the PSO algorithm can be applied to find the optimal combination of the proportional, integral and derivative gains of the PID controller. During the application of the PSO, the initial population will be created randomly with Np individuals containing three decision variables: *kp*, *ki* and *kd*. To evaluate the individuals, we inject these gains into the PID controller and measure the parameters of the corresponding system output using the parameters of the TS model. The choice of the cost function is determined by the objective to be achieved. These objectives are determined by the performance defined in the specifications. Typically, this is static error, rise time, stabilization time and maximum allowed exceedance.

In the literature, we can find a multitude of performance indices. Most of these indices are based on the optimization error. The integrated square error (ISE) is given by:

$$J = \int_0^\infty |e^2(t)| dt \tag{32}$$

Another fitness function is integrated time weight square error (ITSE), which is given as follows:

$$J = \int_0^\infty t\left|e^2(t)\right|dt \tag{33}$$

It seems simpler, yet it is hard to get the ideal time response too. Another widely used fitness function is:

$$J = \int_0^\infty \left(w_1|e(t)| + w_2u^2(t)\right)dt \tag{34}$$

where $e(t)$ and $u(t)$ represent, respectively, the system error and the output of controller, the utilization of this second item is to limited energy (**Figure 3**).

Since, more than the error signal, the performance indices in the time domain are also the overshoot ($D\%$), steady-state error (Ess) the settling time (T_s) and the rise time (T_r). The performance criteria must include all these performances. Therefore, we proposed a new objective function and it is described by the following equation:

$$J = \left(1 - \exp\left(-\beta\right)\right)(D\% + Ess) + \exp\left(-\beta\right)(T_s - T_r) \times \int_0^\infty t\left|e^2(t)\right|dt \tag{35}$$

The framework of online parameter self-tuning for nonlinear system based on TS Fuzzy model is depicted in **Figure 3**.

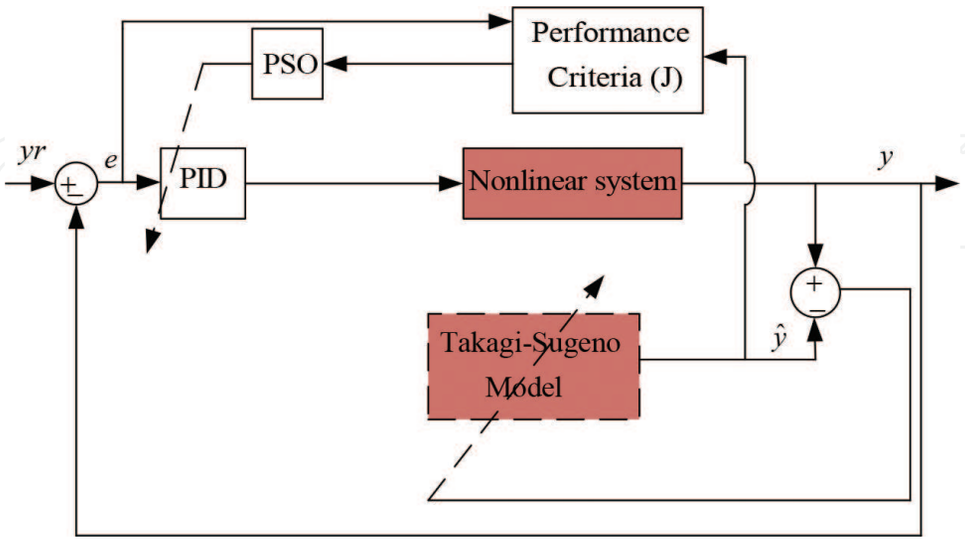

Figure 3. Block diagram of a PID-PSO algorithm.

http://dx.doi.org/10.5772/intechopen.74295

4. Simulation results

This section presents a simulation example to show an application of the proposed control algorithm and its satisfactory performance.

The nonlinear system is characterized by this equation [6]:

$$y(k) = a1^* \sin\left(y(k-1)\right) + a2^* y(k-2) + a3^* u(k-2)^* y(k-3) + b1^* u(k-1) + b2^* \left(\tanh(0.7^* u(k-3))^2\right) \tag{36}$$

with $a1 = 0.4$; $a2 = 0.3$; $a3 = 0.1$; $b1 = 0.6$; and $b2 = 1.8$. Here, $y(k)$ is the output and $u(k)$ is the input which is uniformly bounded in the region $[-2, 2]$.

We choose $[-y(k-1), -y(k-2), u(k-1), u(k-2)]$ as inputs variables, and the number of fuzzy rules is four. The setup applied in this work was the following: the population size was 20, the stopping criterion was 30 generations, $w_{min} = 0.5$, $w_{max} = 0.9$ and $c_1 = c_2 = 2$.

We simulated two experimental cases. In case 1, the simulation result of the control pole placement for the nonlinear system is shown in **Figure 4**.

It can be seen from **Figure 4**, we note that the output of command has an important overshooting. In case 2, the PSO algorithm is adding to PID control. **Figure 5** shows simulation results of the output signal of the control system.

We make a comparative study of PID control by pole placement method and the optimization algorithm of PSO (**Table 1**).

We note though, the advantage of optimizing the parameters of PID controller by PSO compared with the method of pole placement quality control, we observed that the overshoot decreases and the algorithm converges in minimum time.

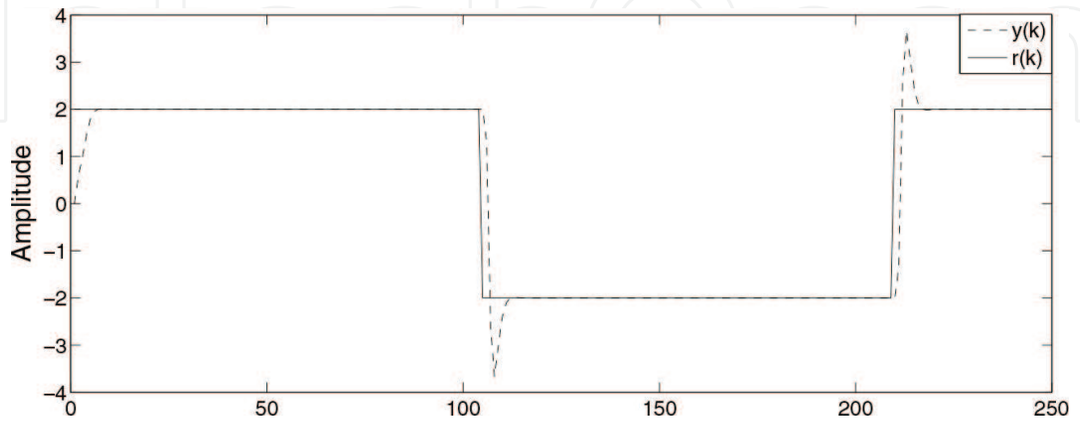

Figure 4. Output signal: PID-pole placement method.

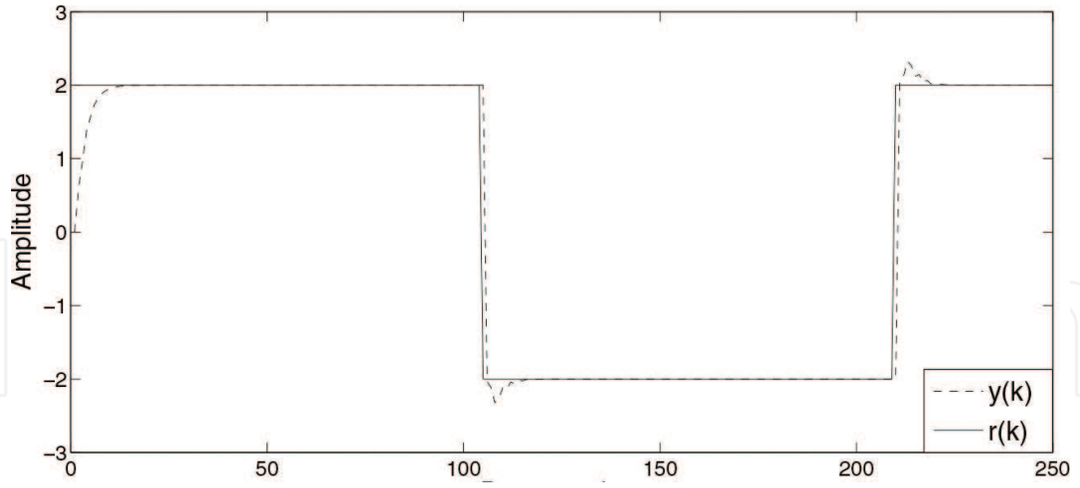

Figure 5. Output signal: PID-optimization of PSO.

	Pole placement method	**Optimization of PSO**
$D(\%)$	64	15
$T_s(s)$	13.568	8.568
$T_r(s)$	10	8

Table 1. Result of PID parameters and PID-PSO.

5. Conclusion

We studied the PID control of a nonlinear system of Takagi-Sugeno for a square input signal using pole placement technique. With this method, we obtain results with an important overshooting. To solve this problem, we have compiled the PID control algorithm with PSO optimization algorithm that it has given good results.

Author details

Taieb Adel* and Chaari Abdelkader

*Address all correspondence to: taeibadel@live.fr

Laboratory of Engineering of Industrial Systems and Renewable Energy, National High School of Engineers of Tunis (ENSIT), Tunis, Tunisia

http://dx.doi.org/10.5772/intechopen.74295

References

[1] Lagrat I, Ouakka H, Boumhidi I. Adaptive control of a class of nonlinear systems based on Takagi-Sugeno fuzzy model. In: Information and Communication Technologies International Symposium (ICTIS'07); Fez, Morocco. 2007. pp. 3-5

[2] Kung CC, Su JY. Affine Takagi-Sugeno fuzzy modelling algorithm by fuzzy cregression models clustering with a novel cluster validity criterion. IET Control Theory Application. 2007;**1**(5):1255-1265

[3] Kennedy J, Eberhart RC. Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks; Perth, Australia. 1995. pp. 1942-1948

[4] Clerc M, Kennedy J. The particle swarm: Explosion, stability, and convergence in multidimension complex space. IEEE Transactions on Evolutionary Computation. 2002;**16**(1):58-73

[5] Aggarwal V, Mao M, O'Reilly U-M. A self-tuning analog proportional-integral-derivative (PID) controller. In: Adaptive Hardware and Systems. 2006. pp. 12-19

[6] Gomm JB, Yu DL, Williams D. A new model structure selection method for non-linear systems in neural modeling. International Conference in Control IEEE Transaction. 1996; **4**(27):752-757

IntechOpen