Journal of Intelligent & Fuzzy Systems xx (20xx) x–xx
DOI:10.3233/JIFS-181014
IOS Press

Intelligent fuzzy controller design for antilock braking systems

Tien-Loc Le[a,b,*]
[a]*Department of Electrical Engineering, Yuan Ze University, Chung-Li, Taoyuan, Taiwan, R.O.C.*
[b]*Department of Electrical Electronic and Mechanical Engineering, Lac Hong University, Bien Hoa, Vietnam*

Abstract. This paper aims to design a self-evolving function-link type-2 fuzzy neural network for application in controlling antilock braking systems. In this control scheme, the self-evolving algorithm is applied to autonomously construct the control network without an initial rule-base. The function-link is designed to give the interval type-2 fuzzy neural network has more freedom in adjusting the parameters. Based on the steepest descent gradient method and the Lyapunov theory, the adaptive laws for the proposed system are derived, and the control system stability is guaranteed. Further, to rapidly achieve the desired control performance, an online particle swarm optimization algorithm is used to optimize the learning rates for the parameter adaptive laws. The performance of the control system is assessed via multiple simulation results of the antilock braking system response under various road conditions.

Keywords: Type-2 fuzzy logic system, particle swarm optimization, self-evolving learning algorithm, antilock braking systems

1. Introduction

Recently, fuzzy controllers and neural networks have attracted many researchers and have been successfully applied in various fields such as control problems, system identification, classification, prediction and medical diagnosis F–6]. The fuzzy system, which expresses the relationship between the antecedent and the consequent by "IF-THEN" rules, uses the linguistic variable and membership functions. Since type-1 fuzzy logic system (T1FLS) was introduced in [7], it has been successfully applied to a wide range of areas. However, owing to the precise sets, T1FLS cannot eliminate the effects of the uncertainties [8]. To overcome this problem, in [9], Zadeh introduced type-2 fuzzy logic systems (T2FLS), which supposed it could cover the uncertainties using the type-2 membership functions (T2MFs) [10]. Indeed, many studies indicate under the same condition, a T2FLS can handle the uncertainties better than a T1FLS [8, 10, 11]. For reducing the computational cost of T2FLS, Liang and Mendel developed the interval type-2 fuzzy logic systems (IT2FLS) [12]. By applying the simple structure, in recent years, the interval type-2 fuzzy neural network (IT2FNN) has attracted the attention of many researchers [13–17]. In 2016, Sumati et al. introduced the parallel interval type-2 subsethood neural fuzzy inference system [13]. Also, in 2016, Li et al. proposed an adaptive sliding mode control for interval type-2 fuzzy systems [15]. In 2017, Herman et al. provided the interval type-2 fuzzy logic system for handling the uncertainty effects in brain-computer interface classification [16]. However, determining a network size for IT2FLS is very important and it significantly influences the control system performance. In [18–21], the authors provided a self-organizing algorithm to self-construct the network size. On the other hand, the self-organizing algorithm also requires the design of an initial structure for the network. To overcome this disadvantage, the self-evolving algorithm was proposed in [22–24], which can self-construct the network size without

*Corresponding author. Tien-Loc Le, E-mail: tienloc@saturn.yzu.edu.tw.

designing the initial structure. However, in the mentioned studies, the learning rates in adaptive laws are fixed and chosen by the trial-and-error method. This study addresses this problem by proposing the PSO algorithm to find the optimal learning rates for self-evolving IT2FNN and also combines with the properties of the functional link network (FLN) to improve the performance.

The functional link network was first introduced by Pao [25], which expanded the network inputs to a higher dimension for exploitating the more striking feature of the original data. The FLN outputs are generated by a nonlinear combination of the original inputs; therefore, it can improve the accuracy of the nonlinear function approximation [26]. Recently, many researchers have applied the FLN to improve the performance of control systems [27–32]. In 2014, Lin et al. provided the SoPC-based function-link cerebellar model articulation control system design for magnetic ball levitation systems [28]. In 2015, Sivachitra and Vijayachitra proposed a metacognitive fully complex-valued functional link network for solving real-valued classification problems [29]. In 2017, Lotfi and Rezaee presented a competitive functional link artificial neural network as a universal approximator [30]. Combining with the functional link network, the self-evolving IT2FNN is referred as a self-evolving function-link type-2 fuzzy neural network (SEFIT2FNN).

The particle swarm optimization (PSO) algorithm is an optimization technique based on mimicking the social behavior of bird flocking or fish schooling to find the global minimum of an objective function [33]. With the ability to solve nonlinear problems and having many advantages such as rapid convergence, as well as simplicity of understanding and implementation, it has been applied in many fields [34]. In recent years, the PSO has often been used to find the optimal parameters for various control systems such as PID, SMC, LQR and neural fuzzy [35–41]. In the adaptive controller, determining the suitable learning rates for adaptive laws is very important and it is highly influential to the system performance. Most previous studies used a trial-and-error method to obtain these values, but it is difficult to achieve the most suitable values and it always takes a long time. Therefore, this study examines the PSO algorithm to obtain the learning rates for updating the weights, the means and the variances of Gaussian membership function in the SEFT2FNN. This network is referred as a PSO-SEFT2FNN.

The proposed controller is applied to control antilock braking systems (ABS). ABS technology was first introduced in 1920 and soon thereafter applied to cars [42]. Nowadays, it has become a safety-critical system in modern cars. The ABS function can maximize the longitudinal tire-road friction while still guaranteeing vehicle steerability. However, the main issues with the ABS design are the highly nonlinear uncertainties in vehicle-braking dynamics, besides the environmental parameters are difficult to identify exactly. Therefore, designing an adaptive ABS system has recently attracted many researchers to various control methods including sliding mode, fuzzy and neural networks. The authors in [26] proposed an intelligent hybrid control system design for antilock braking systems using self-organizing function-link fuzzy cerebellar model articulation controller. The study in [43] introduced the fractional order sliding mode controller design for antilock braking systems. In 2014, Dadashnialehi et al. provided the Intelligent sensorless ABS for in-wheel electric vehicles [42]. In 2015, Wei and Guo introduced an ABS control strategy for commercial vehicle [44]. However, most of these studies fail to achieve satisfactory performance, especially for the transfer between two road conditions.

In the control scheme, the control signal from our controller is directly used to control the ABS system. The studies in [26, 45] designed their networks as uncertainty observers which were used to estimate the lumped uncertainty of the ABS system, and then this uncertainty was used to calculate the control signals for the ideal controller. By using the direct controller, our control system is simpler and easier to apply than the methods in [26, 45]. Compare to previous researches [26, 41–45], this study proposes an effective control method for the ABS system, where it is easy to design the initial network parameters, more freedom in adjusting the variables, and the learning rates are optimized. The major contributions of this study are (i) to develop a self-evolving algorithm for self-constructing the structure of function-link IT2FNN from the empty initial rules base and membership functions; (ii) to design the adaptive laws for updating network parameters; (iii) to optimize the learning rates in adaptive laws using the online and off-line PSO algorithm; (iv) to successfully apply the function-link network, which can give the adaptive laws more freedom in adjusting the variables; (v) to conduct the numerical simulations results of the ABS to illustrate the effectiveness of the proposed control method.

Table 1
The magic formula coefficients

Surface	B	C	D	E
Dry tamac	10	1.9	1	0.97
Wet tamac	12	2.3	0.82	1
Snow	5	2	0.3	1
Ice	4	2	0.1	1

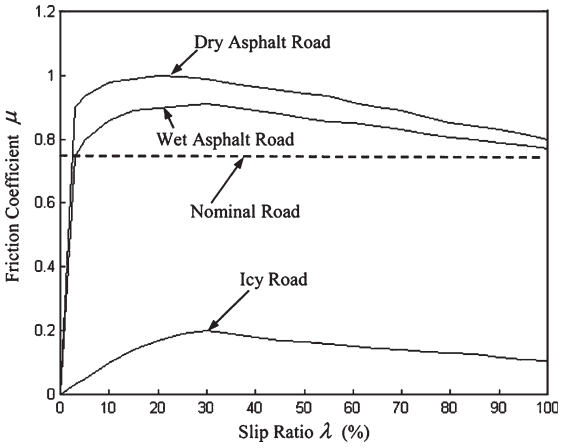


Fig. 1. The relationship between (μ) and (λ) for various road surfaces.

This paper is organized as follows. Section 2 presents the problem formulation of ABS. Section 3 introduces the structure of PSO-SEFT2FNN. The simulation results of the ABS are presented in Section 4. Finally, a conclusion is presented in Section 5.

2. Formulation of ABS

The goal of the ABS controller is control signal generation, which can adjust the wheel slip (λ) to maximize the coefficient of friction (μ) for any given road condition and vehicular speed [26]. During a braking operation, the relation between μ and λ under various road surfaces can be described as the Magic Formula [46].

$$\mu(\lambda) = D.sin\,(C.arctan\,\{B.\lambda - E[B.\lambda - arctan\,(B.\lambda)]\}) \tag{1}$$

where the stiffness, shape, peak, and curvature factors are defined by B, C, D, E. These typical values can be taken in Table 1 [47]. The relation between μ and λ for various road surfaces is shown in Fig. 1 [26].

Referring to [26, 45], the wheel slip equation is defined by:

$$\lambda\,(t) = 1 - \frac{\omega_w\,(t)}{\omega_v\,(t)} \tag{2}$$

where $\omega_v\,(t)$ and $\omega_w\,(t)$ are the angular velocity of the vehicle and the wheel, respectively. Figure 1 shows near the point $\lambda = 20\%$ the μ can get the highest value. Therefore, the target of the controller is generating a control signal that affects the wheel velocity to keep the slip at 20%.

2.1. The wheel dynamics

Applying Newton's law, the dynamic equation of the wheel's angular velocity is determined as [26, 45]

$$\dot{\omega}_w(t) = \frac{1}{J_w}[T_b(t) - B_w\omega_w(t) + T_t(t) \tag{3}$$

where J_w, B_w are the rotational inertia and the viscous friction of the wheel, respectively. $T_b\,(t)$ and $T_t\,(t)$ are the braking torque and the torque generated by the road surface and the wheel.

$$T_t\,(t) = R_w F_t\,(t) \tag{4}$$

where R_w and $F_t\,(t)$ are the radius of the wheel and the tractive force.

$$F_t\,(t) = \mu\,(\lambda)\,N_v\,(\theta) \tag{5}$$

where the nominal reaction force $N_v\,(\theta)$ is given as

$$N_v\,(\theta) = \frac{M_v g}{4} cos\,(\theta) \tag{6}$$

where M_v, θ and g are the mass of the vehicle, the angle of inclination of the road and the gravitational acceleration constant, respectively.

2.2. The vehicle dynamics

The angular velocity of the vehicle ω_v -is defined by the radius of the wheel R_w and the velocity of the vehicle V_v- which is given by

$$\omega_v\,(t) = \frac{V_v\,(t)}{R_w} \tag{7}$$

The acceleration of the vehicle is given as [26, 45] using Newton's law

$$\dot{V}_v(t) = \frac{-1}{M_v}[4F_t(t) + B_v V_v(t) + F_\theta] \tag{8}$$

where B_v and $F_\theta(\theta)$ are the vehicular viscous friction and the force applied to the car, respectively.

$$F_\theta(\theta) = M_v g \sin(\theta) \tag{9}$$

From (2), the derivative of wheel slip is obtained as

$$\dot{\lambda}(t) = \frac{(1-\lambda(t))\dot{\omega}_v - \dot{\omega}_w}{\omega_v} = \frac{-\dot{\omega}_v}{\omega_v}\lambda(t) + \frac{\dot{\omega}_v - \dot{\omega}_w}{\omega_v} \tag{10}$$

From (3), (7), (8) and (10) obtains

$$\dot{\lambda}(t) = f(\lambda;t) + g(t)u(t) \tag{11}$$

where the control effort $u(t)$ is the braking torque $T_b(t)$, and

$$f = \frac{4F_t + B_v R_v \omega_v + F_v}{M_v R_w \omega_v}\lambda - \frac{(4F_t + B_v R_v \omega_v + F_\theta)J_w - (B_w\omega_w - T)M_v R_w}{M_v R_w \omega_v J_w}$$

$$g = \frac{1}{J_w \omega_v}$$

Considering the uncertainties and measurement noise obtains

$$\begin{aligned}\dot{\lambda}(t) &= [f_0(\lambda;t)] + (\lambda;t)] + [g_0(t) + \Delta g(t)]u(t) + d(t) \\ &= f_0(\lambda;t) + g_0(t)u(t) + \beta(\lambda;t)\end{aligned} \tag{12}$$

where $f_0(\lambda;t)$, $g_0(t)$ and $\Delta f(\lambda;t)$, $\Delta g(t)$ denote the nominal parts and the uncertainties of $f(\lambda;t)$, $g(t)$, respectively. $d(t)$ and $\beta\lambda;t$ are the measurement noise and the lumped uncertainty, respectively.

$$\beta(\lambda;t) = \Delta f(\lambda;t) + \Delta g(t)u(t) + d(t) \tag{13}$$

From (12), the ideal controller can be defined as

$$u^*(t) = \frac{1}{g_0(t)}\left[\dot{\lambda}_d(t) - f_0(\lambda;t) - \beta(\lambda;t) + k\lambda_e(t)\right] \tag{14}$$

where $\lambda_e(t)$ is the error between the desired slip trajectory $\lambda_d(t)$ and the wheel slip $\lambda(t)$

$$\lambda_e(t) = \lambda_d(t) - \lambda(t) \tag{15}$$

From (14) and (12) obtains

$$\dot{\lambda}_e(t) + k\lambda_e(t) = 0 \tag{16}$$

If the feedback gain k is assigned by the Hurwitz polynomial approximation, then $\lim_{t\to\infty} \lambda_e(t) = 0$. Since $\beta(\mathbf{x}(t))$ cannot be obtained exactly, the $u^*(t)$ in (14) cannot be obtained. Hence, this study proposed a PSO-SEFT2FNN control system, which is used to mimic the ideal controller $u^*(t)$.

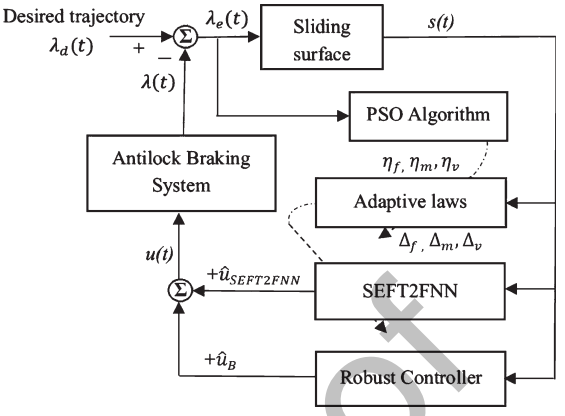


Fig. 2. Block diagram of PSO-SEFT2FNN control system for ABS.

3. PSO-SEFT2FNNC

The block diagram of the ABS control system is shown in Fig. 2, which includes the PSO-SEFT2FNN as the main controller and a robust compensation controller. The sliding surface also is applied to enhance the stability and performance of the control system. Figure 3 shows the structure of the PSO-SEFT2FNN, consisting of the function-link neural network, the IT2FNN and the self-evolving algorithm.

3.1. Structure of function-link network

In this study, the function-link network uses trigonometry to expand the input data (see Fig. 4). Then, the jth output of FLN has applied to calculate the weights for the IT2FNN, which can be obtained by:

$$\begin{aligned}h_j &= f_{j1}\varphi_1 + f_{j2}\varphi_2 \ldots f_j m\varphi_m + \ldots + f_{jn_m}\varphi_{n_m} \\ &= \sum_{m-1}^{n_m} f_{jm}\varphi_m = \boldsymbol{f}_j^T\varphi\end{aligned} \tag{17}$$

where $\boldsymbol{\varphi} = [\varphi_1, \varphi_2, \ldots \varphi_m, \ldots, \varphi_{n_m}]^T$ is the output of FLN, and f_{jm} is the connecting weights between h_j and φ_m, and $m = 1, \ldots, n_m$. For example, considering the FLN with two input, $\mathbf{I} = [i_1, i_2]^T$, then $\boldsymbol{\varphi} = [i_1, \sin(\pi i_1), \cos(\pi i_1), i_2, \sin(\pi i_2), \cos(\pi i_2), i_1 i_2]^T$. The initial value for the connecting weights can be assigned a random value.

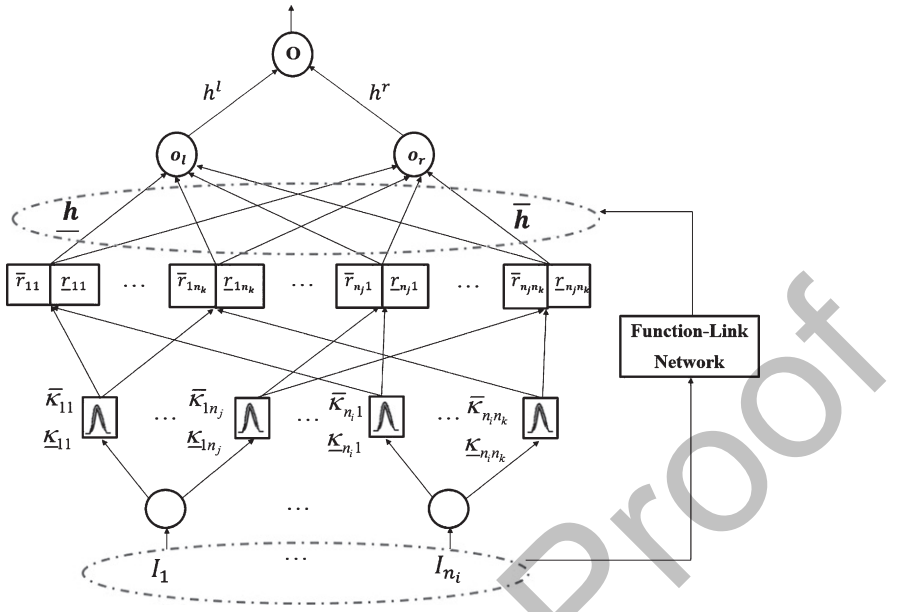


Fig. 3. Structure of the function-link IT2FNN.

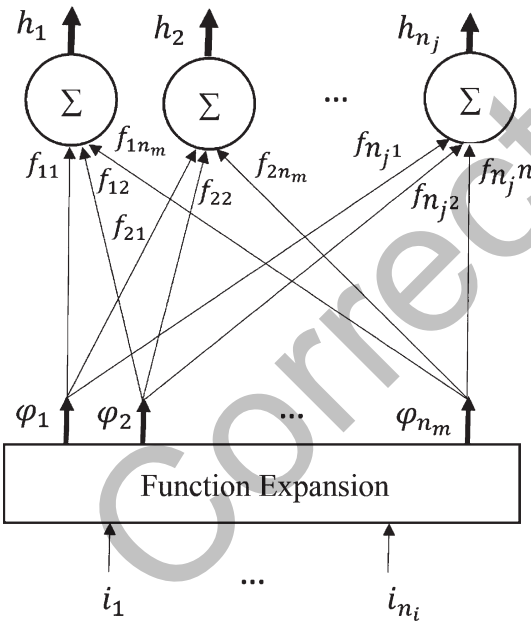


Fig. 4. Structure of the function-link network.

3.2. Structure of function-link IT2FNN

The fuzzy inference rules for the IT2FNN can be shown by the jth rule

$$\text{IF } I_1 \text{ is } \tilde{\kappa}_{1j} \text{ and} \ldots \text{ and } I_i \text{ is } \tilde{\kappa}_{ij} \text{ and} \ldots \text{and } I_n \text{is} \tilde{\kappa}_{nj} \quad \text{THEN } out_j = \tilde{\chi}_j \tag{18}$$

where $\tilde{\kappa}_{ij}$ and $\tilde{\chi}_j$ are the type-2 fuzzy membership functions for the input and output, respectively. $j = 1, \ldots, n_j$ *and* $i = 1, \ldots, n_i$ are denoted the jth rule and the ith fuzzy input.

As shown in Fig. 3, the structure of the IT2FNN has six layers:

1) Input layer: This layer is used to prepare the inputs for the membership function layer and the FNN. All input signals are directly transferred without any computation.
2) Membership function layer: Each node in this layer is a type-2 Gaussian membership function (T2GMF) $\tilde{\kappa}_i j$, which has the fixed mean and uncertainty in the standard deviation. Since $\tilde{\kappa}_{ij}$ is type-2, each output of this layer has two values including: the upper membership function $\bar{\kappa}_i j$ and the lower membership function $\underline{\kappa}_{ij}$

$$\bar{\kappa}_{ij} = \exp\left\{-\frac{1}{2}\left(\frac{I_i - m_{ij}}{\bar{v}_{ij}}\right)^2\right\};$$
$$\underline{\kappa}_{ij} = \exp\left\{-\frac{1}{2}\left(\frac{I_i - m_{ij}}{\underline{v}_{ij}}\right)^2\right\} \tag{19}$$

where $\bar{v}_{ij}$, $\underline{v}_{ij}$ and m_{ij} are the upper variance, lower variance and the mean of the T2GMF, respectively.
3) Firing layer: This layer performs the product operation of all elements in the same block jth. The output of this layer is also an interval value

$R_j = \left[\underline{r}_j, \bar{r}_j\right]$, and can be depicted as:

$$\bar{r}_j = \prod_{j=1}^{n_j} \bar{\kappa}_{ij} \text{ and } \underline{r}_j = \prod_{j=1}^{n_j} \underline{\kappa}_{ij} \tag{20}$$

4) Output weight layer: By using the output of the FLN, as can be seen in (17), the output of this layer is given as:

$$\bar{h} = \begin{bmatrix} \underline{h}_1 \\ \vdots \\ \bar{h}_{n_j} \end{bmatrix} = \begin{bmatrix} \bar{f}_{11} & \cdots & \bar{f}_{1n_m} \\ \vdots & \ddots & \vdots \\ \bar{f}_{n_j1} & \cdots & \bar{f}_{n_jn_m} \end{bmatrix} \begin{bmatrix} \varphi_1 \\ \vdots \\ \varphi_{n_m} \end{bmatrix} = \bar{\boldsymbol{F}}^T \boldsymbol{\varphi} \tag{21}$$

$$\underline{h} = \begin{bmatrix} \underline{h}_1 \\ \vdots \\ \bar{h}_{n_j} \end{bmatrix} = \begin{bmatrix} \underline{f}_{11} & \cdots & \underline{f}_{1n_m} \\ \vdots & \ddots & \vdots \\ \underline{f}_{n_j1} & \cdots & \underline{f}_{n_jn_m} \end{bmatrix} \begin{bmatrix} \varphi_1 \\ \vdots \\ \varphi_{n_m} \end{bmatrix} = \underline{F}^T \boldsymbol{\varphi} \tag{22}$$

5) Pre-output layer: This layer performs the defuzzification operation by using the center of gravity algorithm and the Karnik-Mendel (KM) algorithm in [48].

$$o_l = \frac{\sum_{j=1}^{n_j} r_j^l \underline{h}^j}{\sum_{j=1}^{n_j} r_j^l} \text{ and } o_r = \frac{\sum_{j=1}^{n_j} r_j^r \bar{h}^j}{\sum_{j=1}^{n_j} r_j^r} \tag{23}$$

where r_j^l and r_j^r are given by

$$r_j^l = \begin{cases} \bar{r}_j, & j \le L \\ \underline{r}_j & j > L \end{cases} \text{ and} r_j^r = \begin{cases} \bar{r}_j, & j \le R \\ \underline{r}_j & j > R \end{cases} \tag{24}$$

6) Output layer: This layer performs the average operation between the interval value $[o_l, o_r]$ in the previous layer to obtain the final output.

$$O_{IT2FNN} = \frac{o_l + o_r}{2} \tag{25}$$

3.3. Self-evolving algorithm for FIT2FNN

This section presents the mechanism of the generator and deleting the rule. The first rule (membership function) will be generated based on the first input in the input layer. After that, in the next iteration, the self-evolving algorithm will determine the generation of new rules or delete unnecessary rules. Finally, all the parameters of the existing rules can be updated based on the adaptive law, which is designed in the next section.

The condition for generating the new rule is considered as

$$\text{If } H_g^I < Th_g \text{Then}\{generating\ a\ new\ rule\} \tag{26}$$

$$H_g^I = \max[\kappa_{I1}, \kappa_{I2}, \ldots \kappa_{I1k}] \tag{27}$$

where Th_g and H_g^I are the prior threshold and the maximum membership grade, respectively.

Since the membership grade is the interval value, its average is given by:

$$\kappa_{Ij} = \frac{1}{2}\left(\bar{\kappa}_{Ij} + \underline{\kappa}_{Ij}\right) \tag{28}$$

The parameters for the new rule are given as follows

$$m_{ij}^{N(k)+1=i_i(k)} \tag{29}$$

$$\left[\bar{v}_{ij}^{N(k)+1}, \underline{v}^{N(k)+1}\right] = [v_{init} + \Delta v,\ v_{init} - \Delta v] \tag{30}$$

$$\begin{aligned} \bar{\boldsymbol{f}}_{\boldsymbol{new}}^{N(k)+1} &= \left[f_{init}, f_{init}, \ldots, f_{init}\right]^T \in \Re^m \\ \underline{\boldsymbol{f}}_{new}^{N(k)+1} &= \left[f_{init}, f_{init}, \ldots, f_{init}\right]^T \in \Re^m \end{aligned} \tag{31}$$

where v_{init} and Δv are the initial variance values and its uncertainty term, respectively. The weight and number output of the function-link network are denoted by f_{init} and m. The total number of rules at the kth step is denoted by $N(k)$.

The condition for deleting the unnecessary rule is considered as

$$\text{If } H_d^I < Th_d \text{Then}\{deleting\ unnecessary\ rule\} \tag{32}$$

$$H_d^I = \arg\min[\kappa_{I1}, \kappa_{I2}, \ldots, \kappa_{Ik}] \tag{33}$$

where Th_d and H_d^I are the prior threshold for deleting the rule and the minimum membership grade of the Ith input, respectively.

3.4. The parameter learning algorithm for FIT2FNN

The ideal controller u^* in (14) cannot be obtained because the lumped uncertainty $\beta(\lambda; t)$ is unknown. Therefore, an optimal PSO-SEFT2FNN controller $u^*_{PSO-SEFT2FNN}$ is used to approximate the u^* as:

$$u^*(t) = u^*_{PSO-SEFT2FNN}\left(\underline{f}^*, \bar{f}^*, m^*_{ij}, \underline{v}^*_{ij}, \bar{v}^*_{ij}, t\right) + \in (t) \tag{34}$$

where the approximation error is denoted by $\varepsilon(t)$, and $\underline{\boldsymbol{f}}^*, \bar{\boldsymbol{f}}^*, m_{ij}^*, \underline{v}_{ij}^*, \bar{v}_{ij}^*$ are the ideal parameters of $\underline{\boldsymbol{f}}, \bar{\boldsymbol{f}}, m_{ij}, \underline{v}_{ij}, \bar{v}_{ij}$, respectively. Since the ideal parameters cannot be obtained exactly, an online estimation parameter $\hat{\underline{\boldsymbol{f}}}$, $\hat{\bar{f}}$, $\hat{m}_{ij}$, $\hat{\underline{v}}_{ij}$, $\hat{\bar{v}}_{ij}$ can be designed to obtain the estimation controller $\hat{u}_{PSO-SEFT2FNN}$. Moreover, the robust compensation controller $\hat{u}_B$ is designed for handling the approximation error.

$$\hat{u}(t) = \hat{u}_{PSO-SEFT2FNN}\left(\hat{\underline{\boldsymbol{f}}}, \hat{\bar{f}}, \hat{m}_{ij}, \hat{\underline{v}}_{ij}, \hat{\bar{v}}_{ij}, t\right) + \hat{u}_B(t) \tag{35}$$

where

$$u_B(t) = \hat{B}(t)\mathrm{sgn}(s(t)) \text{ and } \dot{\hat{B}} = \eta_B |s(t)| \tag{36}$$

The high-order sliding mode from [49, 50] is applied to enhance control performance

$$\begin{aligned} s(t) &= \sum_{l=0}^{l-1} \frac{(n-1)!}{l!\,(n-l-1)!}\left(\frac{\partial}{\partial t}\right)^{n-l-1} \zeta^l e \\ &= \lambda_e^{(n-1)} + (n-1)\,\zeta\lambda_e^{(n-2)} + (n-2)\,\zeta^2\lambda_e^{(n-3)} \ldots + \zeta^{n-1}\lambda_e \end{aligned} \tag{37}$$

where ζ and n are a positive constant and the order of sliding surface, respectively.

Consider the change of (37)

$$\begin{aligned} \dot{s}(t) &= \lambda_e^{(n)} + (n-1) + \zeta\lambda_e^{(n-1)} + (n-2)\,\zeta^2\lambda_e^{(n-2)} \ldots + \zeta^{n-1}\lambda_e^1 \\ &= \lambda_e^n + \boldsymbol{K}^T\boldsymbol{\lambda}_e \end{aligned} \tag{38}$$

where the positive gain vector is defined as $\boldsymbol{K} = [(n-1)\zeta, (n-2)\zeta^2, \ldots \zeta^{n-1}]^T \in R^{n-1}$. The values for n and ζ are chosen as the coefficients of a Hurwitz polynomial.

Define the Lyapunov cost function as

$$V(s(t)) = \frac{1}{2}s^2(t), \text{ then } \dot{V}(s(t)) = s(t)\,\dot{s}(t) \tag{39}$$

Apply (12), (15) and (38) into (39) to obtain

$$\begin{aligned} \dot{V}(s(t)) = s(t)\,[&\dot{\lambda}_d(t) - f_0(\lambda; t) - g_0(t)\,(\hat{u}_{PSO-SEFT2FNN}(t) \\ &+ \hat{u}_R(t)) - \beta(\lambda; t) + \zeta\lambda_e(t)] \end{aligned} \tag{40}$$

Apply the gradient descent method and the chain rule, to obtain the online update laws for parameters $\hat{\underline{\boldsymbol{f}}}$, $\hat{\bar{f}}$, $\hat{m}_{ij}$, $\hat{\underline{v}}_{ij}$, $\hat{\bar{v}}_{ij}$ as:

$$\begin{aligned} \hat{\underline{f}}^i(t+1) &= \hat{\underline{f}}^i(t) - \hat{\eta}_f \frac{\partial s(t)\dot{s}(t)}{\partial \hat{\underline{f}}^i} \\ &= \hat{\underline{f}}^i(t) + \frac{1}{2}\hat{\eta}_f s(t)\, g_0(t)\, \frac{r_j^l}{\sum_{j=1}^{n_j} r_j^l}\boldsymbol{\varphi} \end{aligned} \tag{41}$$

$$\begin{aligned} \hat{\bar{f}}(t+1) &= \hat{\bar{f}}(t) - \hat{\eta}_f \frac{\partial s(t)\dot{s}(t)}{\partial \hat{\bar{f}}} \\ &= \hat{\bar{f}}(t) + \frac{1}{2}\hat{\eta}_f s(t)\, g_0(t)\, \frac{r_j^r}{\sum_{j=1}^{n_j} r_j^r}\boldsymbol{\varphi} \end{aligned} \tag{42}$$

$$\begin{aligned} \hat{m}_{ij}(t+1) &= \hat{m}_j^i(t) - \hat{\eta}_m \frac{\partial s(t)\dot{s}(t)}{\partial \hat{m}_{ij}} \\ &= \hat{m}_j^i(t) + \frac{1}{2}\hat{\eta}_m s(t) g_0(t)\left(\frac{(\underline{h}^i - y_l)}{\sum_{j=1}^{n_j} r_j^l}\frac{\partial r_j^l}{\partial \hat{m}_{ij}} + \frac{(\bar{h}^i - y_r)}{\sum_{j=1}^{n_j} r_j^r}\frac{\partial r_j^r}{\partial \hat{m}_{ij}}\right) \end{aligned} \tag{43}$$

$$\begin{aligned} \hat{\bar{v}}_{ij}(t+1) &= \hat{\bar{v}}_{ij}(t) - \hat{\eta}_v \frac{\partial s(t)\dot{s}(t)}{\partial \hat{\bar{v}}_{ij}} \\ &= \hat{\bar{v}}_{ij}(t) + \frac{1}{2}\hat{\eta}_v s(t)\, g_0(t)\left(\frac{(\underline{h}^i - y_l)}{\sum_{j=1}^{n_j} r_j^l}\frac{\partial r_j^l}{\partial \hat{\bar{v}}_{ij}} + \frac{(\bar{h}^i - y_r)}{\sum_{j=1}^{n_j} r_j^r}\frac{\partial r_j^r}{\partial \hat{\bar{v}}_{ij}}\right) \end{aligned} \tag{44}$$

$$\begin{aligned} \hat{\underline{v}}_{ij}(t+1) &= \hat{\underline{v}}_{ij}(t) - \hat{\eta}_v \frac{\partial s(t)\dot{s}(t)}{\partial \hat{\underline{v}}_{ij}} \\ &= \hat{\underline{v}}_{ij}(t) + \frac{1}{2}\hat{\eta}_v s(t)\, g_0(t)\left(\frac{(\underline{h}^i - y_l)}{\sum_{j=1}^{n_j} r_j^l}\frac{\partial r_j^l}{\partial \hat{\underline{v}}_{ij}} + \frac{(\bar{h}^i - y_r)}{\sum_{j=1}^{n_j} r_j^r}\frac{\partial r_j^r}{\partial \hat{\underline{v}}_{ij}}\right) \end{aligned} \tag{45}$$

where the learning-rates $\hat{n}_f, \hat{n}_m, \hat{n}_n$ can be obtained by the PSO algorithm, which is presented in Section 3.5. The terms r_j^l and r_j^r in (41–45) can be $\underline{r}_j$ or $\bar{r}_j$.

$$\frac{\partial \underline{r}_j}{\partial \hat{m}_{ij}} = \frac{\partial \underline{r}_j}{\partial \underline{\kappa}_j^i}\frac{\partial \underline{\kappa}_j^i}{\partial \hat{m}_{ij}} = \underline{r}_j \frac{x_j - \hat{m}_{ij}}{\left(\hat{\underline{v}}_{ij}\right)^2};\ \frac{\partial \bar{r}_j}{\partial \hat{m}_{ij}} = \frac{\partial \bar{r}_j}{\partial \bar{\kappa}_j^i}\frac{\partial \bar{\kappa}_j^i}{\partial \hat{m}_{ij}} = \bar{r}_j \frac{x_j - \hat{m}_{ij}}{\left(\hat{\bar{v}}_{ij}\right)^2} \tag{46}$$

$$\frac{\partial \underline{r}_j}{\partial \hat{\bar{v}}_{ij}} = \frac{\partial \underline{r}_j}{\partial \underline{\kappa}_j^i}\frac{\partial \underline{\kappa}_j^i}{\partial \hat{\bar{v}}_{ij}} = \underline{r}_j \frac{\left(x_j - \hat{m}_{ij}\right)^2}{\left(\hat{\bar{v}}_{ij}\right)^3};\ \frac{\partial \bar{r}_j}{\partial \hat{\bar{v}}_{ij}} = \frac{\partial \bar{r}_j}{\partial \bar{\kappa}_j^i}\frac{\partial \bar{\kappa}_j^i}{\partial \hat{\bar{v}}_{ij}} = \bar{r}_j \frac{\left(x_j - \hat{m}_{ij}\right)^2}{\left(\hat{\bar{v}}_{ij}\right)^3} \tag{47}$$

$$\frac{\partial \underline{r}_j}{\partial \hat{\underline{v}}_{ij}} = \frac{\partial \underline{r}_j}{\partial \underline{\kappa}_j^i}\frac{\partial \underline{\kappa}_j^i}{\partial \hat{\underline{v}}_{ij}} = \underline{r}_j \frac{\left(x_j - \hat{m}_{ij}\right)^2}{\left(\hat{\underline{v}}_{ij}\right)^3};\ \frac{\partial \bar{r}_j}{\partial \hat{\underline{v}}_{ij}} = \frac{\partial \bar{r}_j}{\partial \bar{\kappa}_j^i}\frac{\partial \bar{\kappa}_j^i}{\partial \hat{\underline{v}}_{ij}} = \bar{r}_j \frac{\left(x_j - \hat{m}_{ij}\right)^2}{\left(\hat{\underline{v}}_{ij}\right)^3} \tag{48}$$

By applying the adaptive laws for tuning parameters in (41–45), the PSO-SEFT2FNN controller can achieve the asymptotic stability.

Proof the convergence: Consider the change of the Lyapunov function in (39):

$$\dot{V}(s(t)) = s(t)\,\dot{s}(t) \tag{49}$$

$$\text{Defined } \boldsymbol{\Psi}_z(t) = \frac{\partial \hat{y}_{PSO\text{-}SET2FBELC}}{\partial z}, \ \textit{where } z = \underline{\hat{f}}, \hat{\bar{f}}, \hat{\boldsymbol{m}}, \underline{\hat{\boldsymbol{v}}}, \hat{\bar{\boldsymbol{v}}} \tag{50}$$

Apply the gradient descent method into (49) to obtain

$$\dot{V}(s(t+1)) = \dot{V}(s(t)) + \Delta\dot{V}(s(t)) \cong \dot{V}(s(t)) + \left[\frac{\partial \dot{V}(s(t))}{\partial z}\right]^T \Delta z \tag{51}$$

where Δz and $\Delta\dot{V}(s(t))$ are the change in z and $\dot{V}(s(t))$, respectively.

Applying the chain rule, yields

$$\begin{aligned}\frac{\partial \dot{V}(s(t))}{\partial z} &= \frac{\partial \dot{V}(s(t))}{\partial \hat{y}_{PSO-SEFT2FNN}} \frac{\partial \hat{y}_{PSO-SEFT2FNN}}{\partial z} \\ &= \frac{\partial s(t)\dot{s}(t)}{\partial \hat{y}_{PSO-SEFT2FNN}} \frac{\partial \hat{y}_{PSO-SEFT2FNN}}{\partial z}\end{aligned} \tag{52}$$

From (40) and (51), we have

$$\frac{\partial \dot{V}(s(t))}{\partial z} = -s(t)\frac{\partial \hat{y}_{PSO-SEFT2FNN}}{\partial z} = -s(t)\boldsymbol{\Psi}_z(t) \tag{53}$$

From (41–45), one obtains

$$\Delta z = -\hat{\eta}_z \frac{\partial s(t)\dot{s}(t)}{\partial z} = \hat{\eta}_z s(t)\boldsymbol{\Psi}_z(t) \tag{54}$$

Using (53), (54) and (51), we have

$$\begin{aligned}\Delta\dot{V}(s(t)) &= \left[\frac{\partial \dot{V}(s(t))}{\partial z}\right]^T \Delta z = \left[-s(t)\boldsymbol{\Psi}_z(t)\right]^T \hat{\eta}_z s(t)\boldsymbol{\Psi}_z(t) \\ &= -s^2(t)\hat{\eta}_z \boldsymbol{\Psi}_z(t)\end{aligned} \tag{55}$$

In (55), it is obviousif $\hat{\eta}_z$ is chosen as a positive value then $\Delta\dot{V}$ is negative. Hence, the stability of the system is guaranteed according to the Lyapunov theory.

3.5. Particle swarm optimization

This section presents the application of PSO algorithm to optimize the learning rates $\hat{\eta}_f, \hat{\eta}_m, \hat{\eta}_n$. The flowchart for the PSO algorithm is shown in Fig. 5, and it can be presented by the following steps.

Step 1. Randomly initialize n_p to set the values of $\hat{\eta}_f, \hat{\eta}_m, \hat{\eta}_v$.

Step 2. In turn, run the control system with each set learning rate and then calculate the fitness function.

$$fitness = \exp\left(\frac{\lambda_e(t)^2}{\Gamma_p^2}\right) \tag{56}$$

where Γ_p is a precise fitness function.

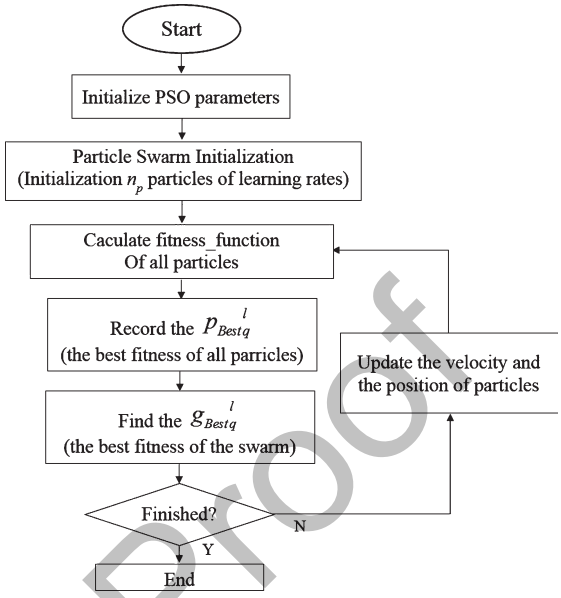


Fig. 5. The PSO flowchart for obtaining the optimal learning rates.

Step 3. Based on the fitness function of the particles and the swarm, choose the best position of the particle ($p_{Best\,q}^{l}$) and the best position of the swarm ($g_{Best\,q}^{l}$)

Step 4. Exit the PSO algorithm if reachese the maximum iteration. Otherwise, go to the next step.

Step 5. Modify all the particles by using the adaptive law given in [39], and then go to Step 2.

$$p_q^l(n+1) = p_q^l(n) + v_q^l(n+1) \tag{57}$$

$$\begin{aligned}v_q^l(n+1) = v_q^l(n) &+ C_1 * \beta_1 * \left[p_{Best\,q}^l - p_q^l(n)\right] \\ &+ C_2 * \beta_2 * \left[g_{Best\,q}^l - p_q^l(n)\right]\end{aligned} \tag{58}$$

where β_1, β_2 and C_1, C_2 are the random variables and the positive acceleration factors, respectively.

4. Simulation results

In this section, the simulation results of ABS control system are considered under three road condition cases. The goal of the control system is to generate the braking torque, which can force the wheel slip $\lambda(t)$ to track the desired slip trajectory $\lambda_d(t)$, meaning the vehicle can achieve the maximum value of the tractive forces. The parameters of the vehicle are shown in Table 2. The desired slip trajectory is given by [26, 45] $\dot{\lambda}_d(t) = -10\lambda_d(t) + 10\lambda_c(t)$. To achieve

Table 2
The vehicle parameters

Parameters	Value
$M_v(kg)$	4×342
$B_v(Ns)$	6
$J_w(N.m.s^2)$	1.13
$R_w(m)$	0.33
$B_w(Ns)$	4
$g(m/s^2)$	9.8

Table 3
Comparison results in RMSE

	Case 1	Case 2	Case 3
Neural Network hybrid [45]	0.0049	0.0173	0.0168
Intelligent hybrid [26]	0.0005	0.0023	0.0010
SOT2FNN [41]	0.00034	0.00025	0.00096
SEFT2FNN	0.00042	0.00034	0.00098
PSO-SEFT2FNN	0.00028	0.00019	0.00087

the maximum slip ratio, $\lambda_c(t)$ is chosen as 0.2 (see Fig. 1). From (2), the term $\lambda(t)$ will go to infinity when $\omega_v(t)$ goes to zero. Hence, in all simulation cases, we consider the control system until the vehicle velocity achieves 5 m/s. To show the effectiveness of the proposed control system, the comparison root mean square error (RMSE) is shown in Table 3.

Initially, the rule base in the PSO-SEFT2FNN is empty. All rules can be auto-generated by using the self-evolving algorithm. The initial parameters as $n_p = 20$, $n_d = 3$, $c1 = c2 = 0.07$, $n = 3$, $\lambda = 0.05$, $v_{init} = 0.5$, $\Delta v = 0.1$, $Th_g = 0.1$ and $Th_d = 0.05$.

Case 1: The dry asphalt road

This case considers the braking action occurring on a dry asphalt road. Assume when the braking action is applied, the vehicle velocity is 25 m/s. Figure 6 presents the ABS response by using the SEFT2FNN without a PSO algorithm. Figure 6(a) shows the angular velocity of the wheel and the vehicle, respectively. Figure 6(b) shows the control force applied to the ABS system. Figure 6(c) presents the slip of the system and the slip reference, respectively. Figures 7–9 show the ABS response using the PSO-SEFT2FNN. Figure 7(a) shows the angular velocity of the wheel and the vehicle, respectively. The control force and the slip of the ABS system are shown in Fig. 7(b) and 7(c), respectively. From Fig. 8, it is obvious the PSO algorithm can rapidly optimize the learning rates for the controller. The change of membership function is shown in Fig. 9.

Case 2: The icy road

This case considers the braking action occurring on an icy asphalt road. Assume when braking action is

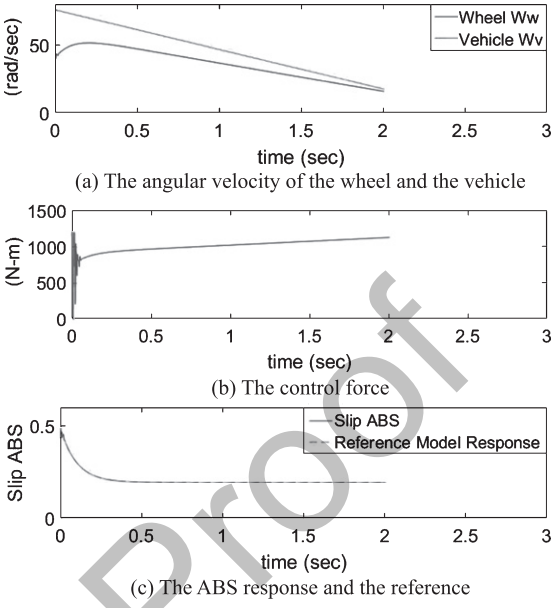


Fig. 6. The ABS response using the SEFT2FNN.

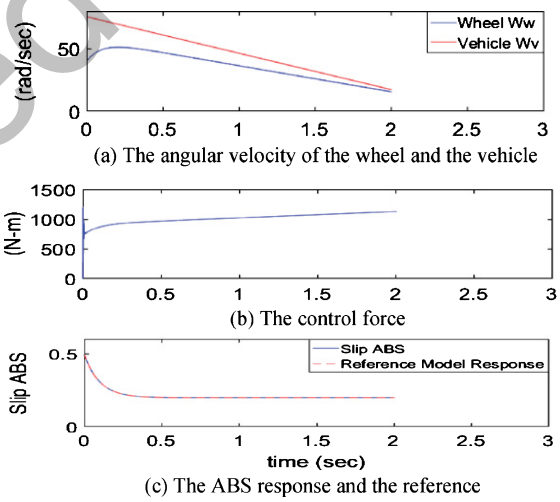
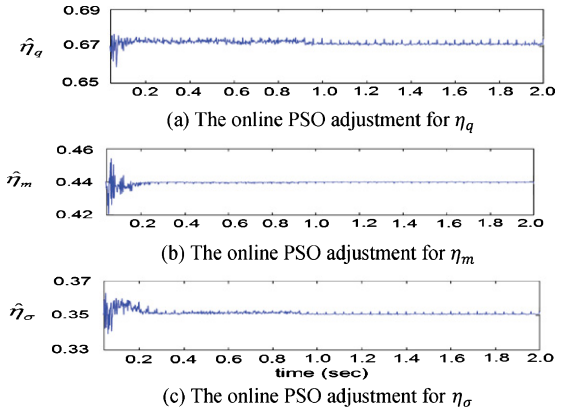


Fig. 7. The ABS response using the PSO-SEFT2FNN.

applied, the vehicle velocity is 12.5 m/s. Figure 10 gives the ABS response by using the SEFT2FNN without a PSO algorithm. Figure 10(a) shows the angular velocity of the wheel and the vehicle, respectively. Figure 10(b) and 10(c) givethe control force and the slip of the ABS system, respectively.TheABS response using the PSO-SEFT2FNN is shown in Figs. 11–13. Figure 11(a) shows the angular velocity of the wheel and the vehicle, respectively. The control force and the slip of the ABS system are shown in Fig. 11(b) and 11(c), respectively.The online PSO

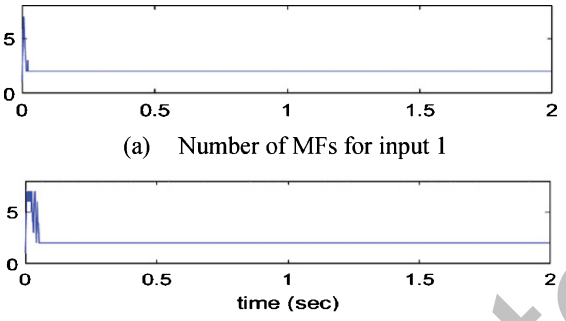


(a) The online PSO adjustment for η_q

(b) The online PSO adjustment for η_m

(c) The online PSO adjustment for η_σ

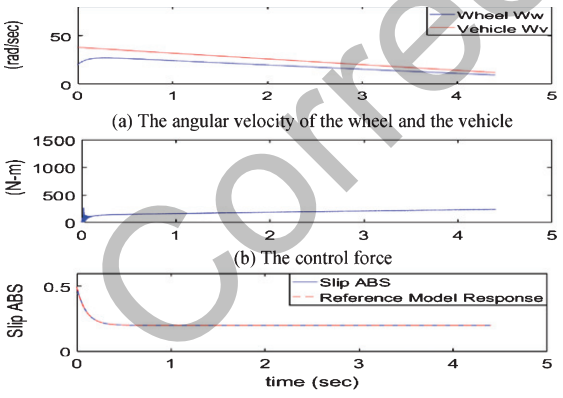
Fig. 8. The online PSO learning-rates adjustment.



(a) Number of MFs for input 1

(b) Number of MFs for input 2

Fig. 9. The number of MFs using the PSO-SEFT2FNN.



(a) The angular velocity of the wheel and the vehicle

(b) The control force

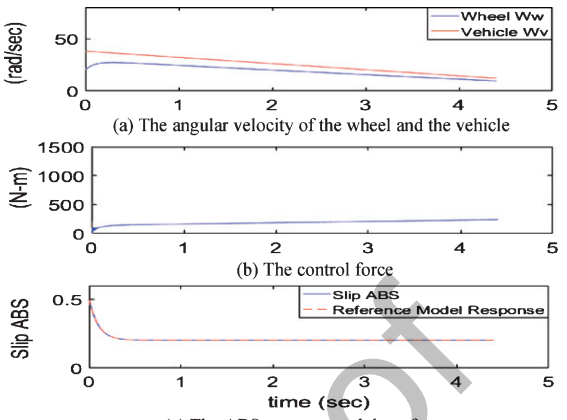
(c) The ABS response and the reference

Fig. 10. The ABS response using the SEFT2FNN.

learning-rates adjustment and the change of membership function are shown in Figs. 12 and 13, respectively.

Case 3: Wet asphalt road to icy road

This case considers the braking action occurring when the vehicle transfers from a wet asphalt road to

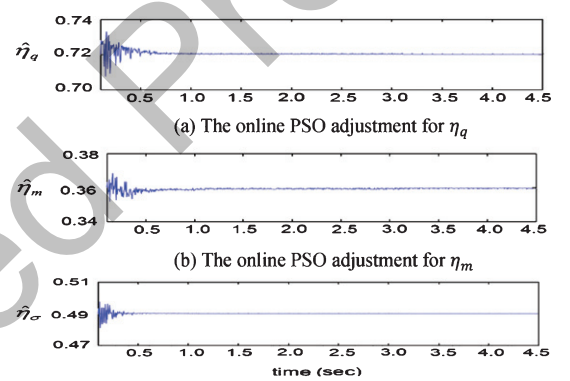


(a) The angular velocity of the wheel and the vehicle

(b) The control force

(c) The ABS response and the reference

Fig. 11. The ABS response using the PSO-SEFT2FNN.



(a) The online PSO adjustment for η_q

(b) The online PSO adjustment for η_m

(c) The online PSO adjustment for η_σ

Fig. 12. The online PSO learning-rates adjustment.

an icy road. Assume when braking action is applied, the vehicle velocity is 12.5 m/s. Figure 14 presents the ABS response by using the SEFT2FNN without PSO algorithm. The angular velocity of the wheel and the vehicle are shown in Fig. 14(a) and 14(b) shows the control force applied to the ABS system. Figure 14(c) presents the slip of the system and the slip reference, respectively. Figures 15–17 show the ABS response using the PSO-SEFT2FNN. Figure 15(a) shows the angular velocity of the wheel and the vehicle, respectively. The control force and the slip of the ABS system are shown in Fig. 15(b) and 15(c), respectively. The online PSO learning-rates adjustment are shown in Figs. 16 and 17 shows the number of the membership function rapidly increases when the road condition changes and after that, it will converge to the optimal number of rules.

In all three cases, it is obvious the PSO algorithm can help the controller rapidly achieve the optimal the learning rates, so the control signal of the PSO-

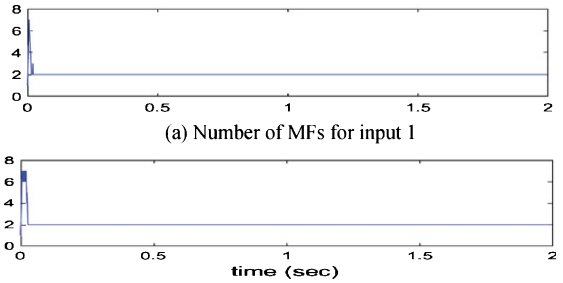


Fig. 13. The number of MFs using the PSO-SEFT2FNN.

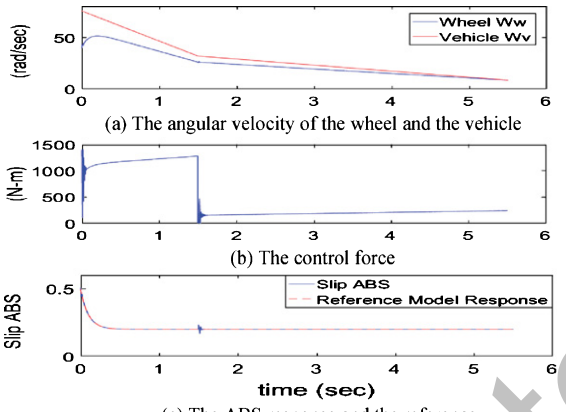


Fig. 14. The ABS response using the SEFT2FNN.

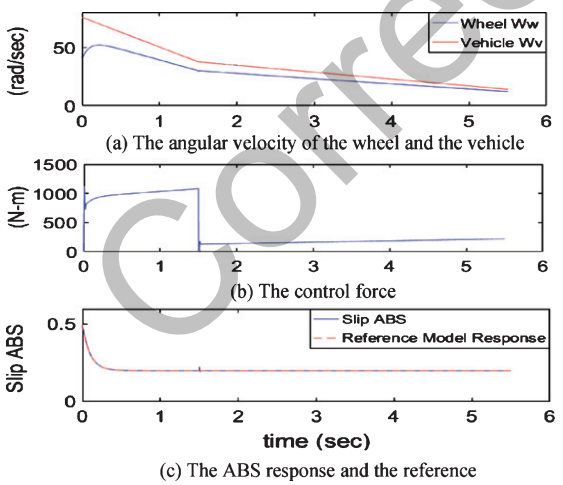


Fig. 15. The ABS response using the PSO-SEFT2FNN.

SEFT2FNN is smoother than the SEFT2FNN without PSO. Figures 9, 13 and 17 show, at the beginning, the controller without any rule can rapidly generate the rules, and then, the number of rules can be

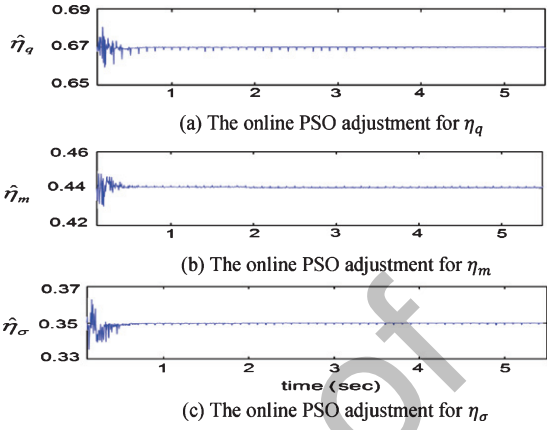


Fig. 16. The online PSO learning-rates adjustment.

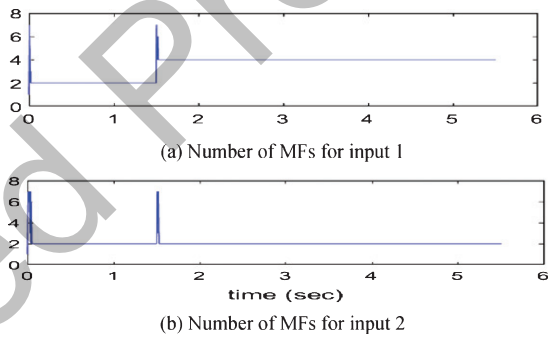


Fig. 17. The number of MFs using the PSO-SEFT2FNN.

optimized by the self-evolving algorithm to achieve better control performance. To limit the number of rules, the maximum membership function for each inputislimited to seven MFs. In the control scheme, the off-line PSO algorithm is first applied to find the optimal learning rates for the proposed controller. After obtaining the optimal learning rates and suitable parameters, the control system will be controlled with the online-PSO SEFIT2FNN to rapidly cope with the changes. During the control process, the parameters can on-line update to the suitable value by using the adaptive laws which are described in Section 3.3.

Choosing the threshold for generating and deleting the rules affects much of the control system. If the generation threshold is large, it will lead to fewer rules being generated. Contrarily, if the generating threshold is very small, a huge number of rules are generated. If the deleting threshold is large, it will barely delete any rules. Contrarily, if the detecting threshold is very small, many rules are deleted and just a few rules remain. Having only a few rules may cause the control system to achieve suboptimal per-

formance, whereas a huge number of rules leads to a large computation time. So choosing these thresholds by try-and-error is very important.

5. Conclusion

This study proposes a PSO-SEFT2FNN control system that can achieve favorable control performance for the antilock braking system. Numerous simulations are conducted under various road conditions to verify the effectiveness of the control system. The main contributions of this study are: successful design of a self-evolving algorithm for self-constructing the structure of function-link IT2FNN from the empty initial rules base and membership functions; successfully design of adaptive laws for updating network parameters; the learning rates in adaptive laws are optimized using the online and off-line PSO algorithm; successfully application of the function-link network, which can give the adaptive laws more freedom in adjusting the variables; the system's stability is guaranteed by the Lyapunov function. Application of the estimation method to estimate the system state and deal with the noise disturbances will be our future work. Finally, beside the application for the control system, the proposed method is also suitable for other applications such as system identification, classification, and prediction.

References

[1] O. Erkaymaz, M. Ozer and M. Perc, Performance of small-world feedforward neural networks for the diagnosis of diabetes, *Applied Mathematics and Computation* **311** (2017), 22–28.

[2] I. Fister, P.N. Suganthan, S.M. Kamal, F.M. Al-Marzouki, M. Perc and D. Strnad, Artificial neural network regression as a local search heuristic for ensemble strategies in differential evolution, *Nonlinear Dynamics* **84**(2) (2016), 895–914.

[3] C.-M. Lin, V.-H. La and T.-L. Le, DC–DC converters design using a type-2 wavelet fuzzy cerebellar model articulation controller, *Neural Computing and Applications* (2018). doi: 10.1007/s00521-018-3755-z

[4] G. Wei, F. E. Alsaadi, T. Hayat and A. Alsaedi, Hesitant bipolar fuzzy aggregation operators in multiple attribute decision making, *Journal of Intelligent & Fuzzy Systems* **33**(2) (2017), 1119–1128.

[5] E. Karakose, M.T. Gencoglu, M. Karakose, O. Yaman, I. Aydin and E. Akin, A new arc detection method based on fuzzy logic using S-transform for pantograph–catenary systems, *Journal of Intelligent Manufacturing* **29**(4) (2018), 839–856.

[6] H. Zhang, J. Hu, L. Zou, X. Yu and Z. Wu, Event-based state estimation for time-varying stochastic coupling networks with missing measurements under uncertain occurrence probabilities, *International Journal of General Systems* **47**(5) (2018), 506–521.

[7] L.A. Zadeh, Fuzzy sets, *Information and Control* **8** (1965), 338–353.

[8] O. Castillo, R. Martínez-Marroquín, P. Melin, F. Valdez and J. Soria, Comparative study of bio-inspired algorithms applied to the optimization of type-1 and type-2 fuzzy controllers for an autonomous mobile robot, *Information Sciences* **192** (2012), 19–38.

[9] L.A. Zadeh, The concept of a linguistic variable and its application to approximate reasoning—I, *Information Sciences* **8**(3) (1975), 199–249.

[10] J.M. Mendel, A quantitative comparison of interval type-2 and type-1 fuzzy logic systems: First results, *Proceedings IEEE International Conference on Fuzzy Systems*, 2010, pp. 1–8.

[11] S.-K. Oh, H.-J. Jang and W. Pedrycz, A comparative experimental study of type-1/type-2 fuzzy cascade controller based on genetic algorithms and particle swarm optimization, *Expert Systems with Applications* **38**(9) (2011), 11217–11229.

[12] Q. Liang and J.M. Mendel, Interval type-2 fuzzy logic systems: Theory and design, *IEEE Transactions on Fuzzy Systems* **8**(5) (2000), 535–550.

[13] V. Sumati, P. Chellapilla, S. Paul and L. Singh, Parallel interval type-2 subsethood neural fuzzy inference system, *Expert Systems with Applications* **60** (2016), 156–168.

[14] H.M. Nehi and A. Keikha, TOPSIS and Choquet integral hybrid technique for solving MAGDM problems with interval type-2 fuzzy numbers, *Journal of Intelligent & Fuzzy Systems* **30**(3) (2016), 1301–1310.

[15] H. Li, J. Wang, H.-K. Lam, Q. Zhou and H. Du, Adaptive sliding mode control for interval type-2 fuzzy systems, *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **46**(12) (2016), 1654–1663.

[16] P.A. Herman, G. Prasad and T.M. McGinnity, Designing an interval type-2 fuzzy logic system for handling uncertainty effects in brain–computer interface classification of motor imagery induced EEG patterns, *IEEE Transactions on Fuzzy Systems* **25**(1) (2017), 29–42.

[17] T. Kumbasar, I. Eksin, M. Guzelkaya and E. Yesil, Type-2 fuzzy model based controller design for neutralization processes, *ISA Transactions* **51**(2) (2012), 277–287.

[18] N. Wang, M.J. Er and X. Meng, A fast and accurate online self-organizing scheme for parsimonious fuzzy neural networks, *Neurocomputing* **72**(16-18) (2009), 3818–3829.

[19] C.-M. Lin, Y.-M. Chen and C.-S. Hsueh, A Self-organizing interval type-2 fuzzy neural network for radar emitter identification, *International Journal of Fuzzy Systems* **16**(1) (2014), 20–30.

[20] C. Li and C.-Y. Lee, Self-organizing neuro-fuzzy system for control of unknown plants, *IEEE Transactions on Fuzzy Systems* **11**(1) (2003), 135–150.

[21] C.M. Lin and T.Y. Chen, Self-organizing CMAC control for a class of MIMO uncertain nonlinear systems, *IEEE Trans Neural Networks* **20**(9) (2009), 1377–1384.

[22] Y.-Y. Lin, J.-Y. Chang and C.-T. Lin, A TSK-type-based self-evolving compensatory interval type-2 fuzzy neural network (TSCIT2FNN) and its applications, *IEEE Transactions on Industrial Electronics* **61**(1) (2014), 447–459.

[23] S.-Y. Chen and T.-S. Liu, Intelligent tracking control of a PMLSM using self-evolving probabilistic fuzzy neural

network, *IET Electric Power Applications* **11**(6) (2017), 1043–1054.

[24] Y.-T. Liu, Y.-Y. Lin, S.-L. Wu, C.-H. Chuang and C.-T. Lin, Brain dynamics in predicting driving fatigue using a recurrent self-evolving fuzzy neural network, *IEEE Transactions on Neural Networks and Learning Systems* **27**(2) (2016), 347–360.

[25] Y.-H. Pao, Functional link nets: Removing hidden layers, *AI Expert* **4**(4) (1989), 60–68.

[26] C.-M. Lin and H.-Y. Li, Intelligent hybrid control system design for antilock braking systems using self-organizing function-link fuzzy cerebellar model articulation controller, *IEEE Transactions on Fuzzy Systems* **21**(6) (2013), 1044–1055.

[27] C.-H. Chen, C.-J. Lin and C.-T. Lin, A functional-link-based neurofuzzy network for nonlinear system control, *IEEE Transactions on Fuzzy Systems* **16**(5) (2008), 1362–1378.

[28] C.-M. Lin, Y.-L. Liu and H.-Y. Li, SoPC-based function-link cerebellar model articulation control system design for magnetic ball levitation systems, *IEEE Transactions on Industrial Electronics* **61**(8) (2014), 4265–4273.

[29] M. Sivachitra and S. Vijayachitra, A metacognitive fully complex valued functional link network for solving real valued classification problems, *Applied Soft Computing* **33** (2015), 328–336.

[30] E. Lotfi and A.A. Rezaee, A competitive functional link artificial neural network as a universal approximator, *Soft Computing* **22**(14) (2017), 4613–4625.

[31] J.C. Patra and A. Van den Bos, Modeling of an intelligent pressure sensor using functional link artificial neural networks, *ISA Transactions* **39**(1) (2000), 15–27.

[32] B. Zhang and W. Zhang, Adaptive predictive functional control of a class of nonlinear systems, *ISA Transactions* **45**(2) (2006), 175–183.

[33] J. Kennedy and R.C. Eberhart, Particle swarm optimization, *Proceedings IEEE International Conference on Neural Networks***4** (1995), 1942–1948.

[34] J.-J. Kim and J.-J. Lee, Trajectory optimization with particle swarm optimization for manipulator motion planning, *IEEE Transactions on Industrial Informatics* **11**(3) (2015), 620–631.

[35] Z. Bingül and O. Karahan, A fuzzy logic controller tuned with PSO for 2 DOF robot trajectory control, *Expert Systems with Applications* **38**(1) (2011), 1017–1031.

[36] R.-J. Wai, J.-D. Lee and K.-L. Chuang, Real-time PID control strategy for maglev transportation system via particle swarm optimization, *IEEE Transactions on Industrial Electronics* **58**(2) (2011), 629–646.

[37] B. Ufnalski, A. Kaszewski and L.M. Grzesiak, Particle swarm optimization of the multioscillatory LQR for a three-phase four-wire voltage-source inverter with an LC Output Filter, *IEEE Transactions on Industrial Electronics* **62**(1) (2015), 484–493.

[38] V.H. Haji and C.A. Monje, Fractional order fuzzy-PID control of a combined cycle power plant using Particle Swarm Optimization algorithm with an improved dynamic parameters selection, *Applied Soft Computing* **58** (2017), 256–264.

[39] C.-M. Lin and T.-L. Le, WCMAC-based control system design for nonlinear systems using PSO, *Journal of Intelligent & Fuzzy Systems* **33**(2) (2017), 807–818.

[40] H. Melo and J. Watada, Gaussian-PSO with fuzzy reasoning based on structural learning for training a neural network, *Neurocomputing* **172** (2016), 405–412.

[41] C.-M. Lin and T.-L. Le, PSO-self-organizing interval type-2 fuzzy neural network for antilock braking systems, *International Journal of Fuzzy Systems* **19**(5) (2017), 1362–1374.

[42] A. Dadashnialehi, A. Bab-Hadiashar, Z. Cao and A. Kapoor, Intelligent sensorless ABS for in-wheel electric vehicles, *IEEE Transactions on Industrial Electronics* **61**(4) (2014), 1957–1969.

[43] Y. Tang, X. Zhang, D. Zhang, G. Zhao and X. Guan, Fractional order sliding mode controller design for antilock braking systems, *Neurocomputing* **111** (2013), 122–130.

[44] Z. Wei and G. Xuexun, An ABS control strategy for commercial vehicle, *IEEE/ASME Transactions on Mechatronics* **20**(1) (2015), 384–392.

[45] C.M. Lin and C.F. Hsu, Neural-network hybrid control for antilock braking systems, *IEEE Transactions on Neural Networks* **14**(2) (2003), 351–359.

[46] H.B. Pacejka and E. Bakker, The magic formula tyre model, *Vehicle System Dynamics* **21**(S1) (1992), 1–18.

[47] M. Vaezi, E.S.J. Hesam and S. Anwar, IMC-PID traction control system for an automobile via engine torque control, *Proceedings IEEE International Conference on Industrial Technology*, 2015, pp. 296–302.

[48] J. M. Mendel, Uncertain rule-based fuzzy logic systems: Introduction and new directions, *IEEE Computational Intelligence Magazine*, *Prentice-Hall, Upper Saddle River*, 2001.

[49] M. Manceur, N. Essounbouli and A. Hamzaoui, Second-order sliding fuzzy interval type-2 control for an uncertain system with real application, *IEEE Transactions on Fuzzy Systems* **20**(2) (2012), 262–275.

[50] S. Ahmed, N. Shakev, A. Topalov, K. Shiev and O. Kaynak, Sliding mode incremental learning algorithm for interval type-2 Takagi–Sugeno–Kang fuzzy neural networks, *Evolving Systems* **3**(3) (2012), 179–188.