Fuzzy Sets and Systems 132 (2002) 201–216

www.elsevier.com/locate/fss

A new fuzzy neural network with fast learning algorithm and guaranteed stability for manufacturing process control

Yunfei Zhou, Shuijin Li *, Rencheng Jin

National Engineering Center of Numerical Control System, College of Mechanical Science & Engineering, Huazhong University of Science & Technology, Wuhan, Hubei 430074, China

Received 30 June 1999; received in revised form 23 May 2001; accepted 15 October 2001

Abstract

In this paper, a new fuzzy neural network (FNN) is presented for manufacturing process control. It is different from the conventional FNN in its structure, learning algorithm and stability analysis method. Firstly, it utilizes the input and output layer to on-line fine-tune scaling factors. It can also use the hidden layers to realize the fuzzification, fuzzy inference, defuzzification and tune parameters such as membership functions, fuzzy control rules dynamically. Secondly, a new combining learning algorithm (CL) which combines the gradient-based error back-propagation algorithm (EBP) with similar Newton (SN) algorithm is proposed in order to improve the convergence speed and release computational burden during the learning process. Lastly, a convergence condition for determining the stability of FNN is established. Physical experiments for manufacturing process control are implemented to evaluate the effectiveness of the proposed scheme. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Neuro-fuzzy systems; Learning; Lyapunov's direct method; Stability analysis

1. Introduction

Recently, fuzzy neural network control systems have been extensively studied [2,7,22,24,29] and successfully used in manufacturing process control, such as tool wear monitoring [21,30], multi-sensor integration for intelligent control of machining [20], etc.

In general, the design of FNN includes (1) the structure, (2) the learning algorithm and (3) the stability analysis.

Many structures for FNN have been proposed. In [31], a five-layer FNN for learning rules of fuzzy logic control systems was proposed and a two-phase learning procedure was developed to delete redundant rules for obtaining a concise fuzzy rule base. In [4,7,10,22,39], a feedforward multilayer connectionist network is proposed to realize the fuzzification, fuzzy operator and defuzzification. A class of adaptive FNN was developed in [3,13,21,26,35] to on-line adjust membership functions, fuzzy logic rules and so on. Unfortunately, most of these researchers have ignored the scaling factors that may affect the dynamic performance of FNN, so that the designed FNN cannot

* Corresponding author. Fax: +86-27-8754-8737.
E-mail address: hadt@21cn.com (S. Li).

entirely replace the traditional control systems (i.e., PID control system).

A variety of effective learning algorithms based on EBP were developed for FNN or neural networks [4,15,24,25,31]. For instance, a compensatory learning algorithm in [4], a successive overrelaxation back-propagation algorithm in [23], a generalized back-propagation algorithm in [18], a hybrid linear/nonlinear training algorithm for training feedforward neural networks in [27] etc. However, these works have not successfully solved an essential problem, which is that the EBP algorithm converges very slowly near the point with extremum.

Some papers [5,6,8,9,12,15,16,19,28,30,32,40] have studied the stability of FNN. In [19], the stability can be analyzed by the Nyquist stability criterion. In [32], the stability of the T–S model was studied by using a fuzzy block diagram. In [16], an energetic stability algorithm (ESA) was proposed to investigate the local stability of a free dynamic system. In [28], the author developed a numerical stability analysis method, through which we can observe and study the system behavior by a direct computation of the system trajectory, etc. But, there have been few studies on stability analysis by using the convergence of a learning algorithm.

In this paper, a new FNN (shown in Fig. 1) is designed to solve the above-mentioned problems.

Firstly, the proposed FNN can realize the fuzzification, fuzzy inference, defuzzification. The main difference between the proposed FNN and others lies in the layer *A* and layer *G* where the scale conversion can be finished dynamically.

Secondly, the SN algorithm is proved to converge faster than gradient-based EBP algorithm and has lighter computational burden than Newton method near the point with extremum. Based on the SN algorithm, a new CL learning algorithm is proposed to improve the convergence speed and release computational burden during the learning procedure.

Lastly, the fuzzy control rules have been embedded in the neural network, so the stability of FNN can be associated with the convergence of CL learning algorithm used for training neural network. For this reason, extending the work [7,18], we developed a stability analysis method by using the convergence of CL learning algorithm.

This paper is organized as follows. In Section 2, a seven-layer feedforward FNN is presented. In Section 3, a combining learning algorithm (CL) is proposed. A theorem and corollary are proposed and proved in Section 4. In Section 5, physical experiments for manufacturing process control are presented to illustrate the performance of the new FNN. Conclusions are given in Section 6.

2. Structure of the FNN

In this section, we will construct a feedforward seven-layer fuzzy neural network to implement the fuzzy control rules stated in (Eq. (1)). Shown as Fig. 1, each neuron in layer A represents one input valuable. Layer B realizes the fuzzification and layer C can construct membership functions. The links between layer C and layer D are the antecedent links and those between layer E and layer F are the consequent links. Defuzzification can be realized in layer F. Layer G can act as a proportion controller to implement scale conversion and provide the actual control values for plants.

The proposed FNN realizes the following fuzzy control rules:

$$R^l: \quad \text{IF } x_1 \text{ is } A_1^l \ldots x_n \text{ is } A_n^l,$$
$$\text{THEN } y_1 \text{ is } B_1^l \ldots y_m \text{ is } B_m^l, \tag{1}$$

where $l = 1, 2, \ldots, S$ and S denotes the sum of fuzzy control rules.

If we use minimum operator and centroid defuzzifier [36], the fuzzy control rules stated in (Eq. (1)) can be implemented with the following nonlinear mapping equation (from $x \in R^n$ to $y \in R^m$).

$$y_j = \frac{\sum_{l=1}^{S} \pi_j^l \min_n\{\mu_{A_i^l}(x_i)\}}{\sum_{l=1}^{S} \min_n\{\mu_{A_i^l}(x_i)\}}, \quad j = 1, 2, \ldots, m, \tag{2}$$

where π_j^l implies the matching degree of lth fuzzy control rule. A_i^l and B_i^l are fuzzy sets. The $\mu_{A_i^l}$ in Eq. (2) is the membership function of fuzzy set A_i^l.

The semantic meaning and function of the neurons in the proposed fuzzy neural network are as follows.

Layer A (*input*): Each neuron represents one input variable. It just converts the input variable to the next

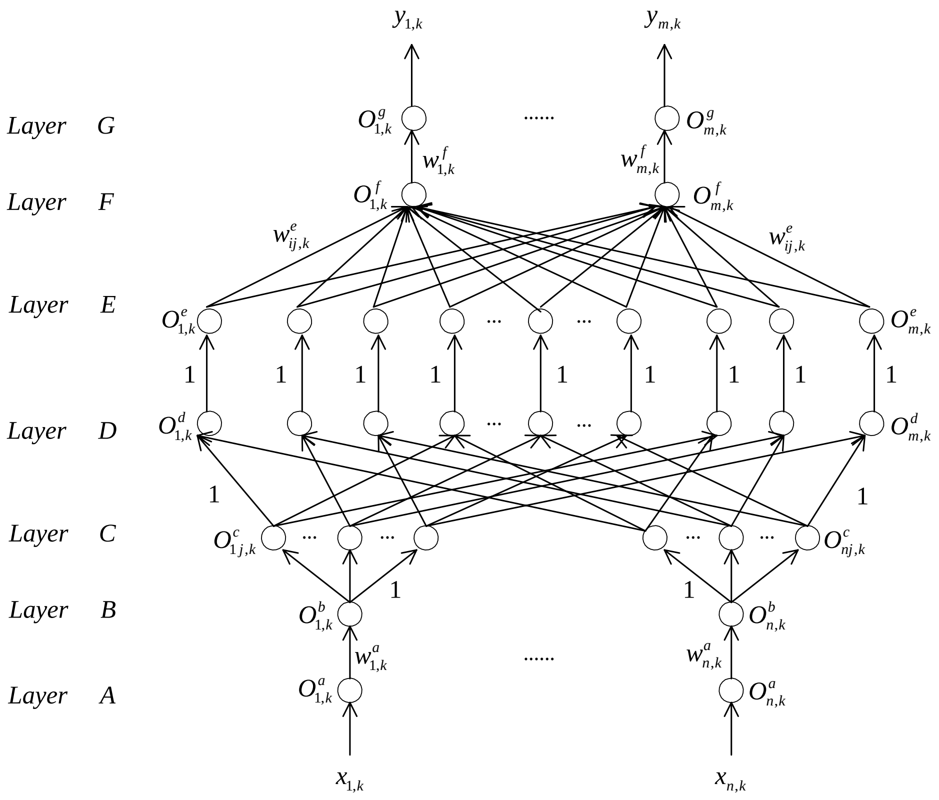

Fig. 1. Structure of the proposed FNN.

layer directly, i.e.,

$$I^a_{i,k} = x_{i,k}, \quad O^a_{i,k} = I^a_{i,k}, \qquad i = 1,2,\dots,n, \tag{3}$$

where $I^a_{i,k}$ and $O^a_{i,k}$ denote, respectively, the input and output of ith neuron in layer A. In addition, k represents the training data number, i.e. $(x_{i,k}, y_{i,k})$.

Layer B (*fuzzification*): Layer B realizes the fuzzification. In this layer, the relation between the input and output is represented as

$$I^b_{i,k} = w^a_{i,k} O^a_{i,k} = w^a_{i,k} x_{i,k}, \quad O^b_{i,k} = I^b_{i,k}, \qquad i = 1,2,\dots,n, \tag{4}$$

where $w^a_{i,k}$ is the link weight (or scaling factor for fuzzification) of the ith neuron.

Layer C (*membership function*): In this layer, each neuron represents the membership function of a linguistic variable. The most commonly used membership functions are in shape of triangle, trapezoid and bell, etc. In this paper, seven fuzzy sets (NB, NM, NS, ZO, PS, PM and PB) are used for the above-mentioned fuzzy control rules, and the membership functions for these seven fuzzy sets are all bell-shaped.

So the output of the jth term neuron associated with $O^b_{i,k}$ is

$$I^c_{ij,k} = -\frac{(O^b_{i,k} - C_{ij,k})^2}{\sigma^2_{ij,k}}, \quad O^c_{ij,k} = e^{I^c_{ijk}}, \quad i = 1,2,\dots,n, \quad i = 1,2,\dots,7, \tag{5}$$

where $C_{ij,k}$ and $\sigma_{ij,k}$ are, respectively, the center and the width of the bell-shaped membership function. In addition, the link weight in layer C is assumed to be unity by default.

Layer D (*rule*): In Eq. (2), we have assumed that the minimum operator is adopted in FNN, so we can get

$$I^d_{i,k} = \min\{O^c_{1i_1,k}, O^c_{2i_2,k}, \ldots, O^c_{ni_n,k}\},$$
$$O^d_{i,k} = I^d_{i,k},$$
$$i = 1,2,\ldots,S, \quad S = \prod_{i=1}^{n} 7 = 7^n. \tag{6}$$

As the layer C, the link weights of layer D are also set to be unity by default.

Layer E (*rule matching*): This layer can calculate the matching degrees of fuzzy control rules.

$$I^e_{i,k} = O^d_{i,k} \Big/ \sum_{i=1}^{s} O^d_{i,k}, \quad i = 1,2,\ldots,S. \tag{7}$$
$$O^e_{i,k} = I^e_{i,k},$$

Layer F (*defuzzification*): The neurons in layer F perform defuzzification.

$$I^f_{i,k} = \sum_{j=1}^{S} w^e_{ij,k} O^e_{j,k}, \quad i = 1,2,\ldots,m. \tag{8}$$
$$O^f_{i,k} = I^f_{i,k},$$

Layer G (*output*): This is the output layer in the proposed FNN. It holds the function of converting the output of fuzzy control rules to actual control values directly inputted into the plants.

$$I^g_{i,k} = w^f_{i,k} O^f_{i,k}, \quad i = 1,2,\ldots,m. \tag{9}$$
$$O^g_{i,k} = I^g_{i,k},$$

Therefore, layer G can be considered as a proportion controller and the link weight $w^f_{i,k}$ corresponds to the scaling gain K_p.

3. A new combining learning (CL) algorithm

In this section, we will describe a new combining learning (CL) algorithm. The objective of proposed learning algorithm is to associate input–output training samples by properly tuning the weights and other parameters of FNN, so that a specific error signal is minimized.

3.1. Gradient-based EBP algorithm

A EBP learning algorithm based on gradient descent is employed here to tune $C_{ij,k}, \sigma_{ij,k}, w^a_{i,k}, w^f_{i,k}$.

The cost function we want to minimize is defined as

$$J_k = \frac{1}{2} E_k^2,$$
$$E_k = \frac{1}{2} \sum_{i=1}^{m} (y_{i,k} - O^g_{i,k})^2 = \frac{1}{2} \sum_{i=1}^{m} e^2_{i,k},$$
$$k = 1,2,\ldots,r, \tag{10}$$

where m is the sum of neurons of output layer (i.e., layer G). The $y_{i,k}$ and $O^g_{i,k}$ are, respectively, the target and actual output of the ith neuron of output layer. Note that r represents the sum of training samples, which are used to train the FNN.

Next, we will derive the learning law for each layer in feedbackward direction.

Layer G: From Eqs. (10) and (9), the error term to be propagated is given as

$$\delta^g_{i,k} = -\frac{\partial J_k}{\partial I^g_{i,k}} = -\frac{\partial J_k}{\partial O^g_{i,k}}$$
$$= E_k (y_{i,k} - O^g_{i,k}) = E_k e_{i,k}, \quad i = 1,2,\ldots,m. \tag{11}$$

Layer F: Similarly to the above definition and from Eqs. (11), (9) and (8), we have

$$\delta^f_{i,k} = -\frac{\partial J_k}{\partial I^f_{i,k}} = -\frac{\partial J_k}{\partial I^g_{i,k}} \frac{\partial I^g_{i,k}}{\partial O^f_{i,k}} \frac{\partial O^f_{i,k}}{\partial I^f_{i,k}}$$
$$= \delta^g_{i,k} w^f_{i,k}, \quad i = 1,2,\ldots,m. \tag{12}$$

Layer E: Using Eqs. (12), (8) and (7), we can get

$$\delta^e_{j,k} = -\frac{\partial J_k}{\partial I^e_{j,k}} = -\sum_{i=1}^{m} \frac{\partial J_k}{\partial I^f_{i,k}} \frac{\partial I^f_{i,k}}{\partial O^e_{j,k}} \frac{\partial O^e_{j,k}}{\partial I^e_{j,k}}$$
$$= \sum_{i=1}^{m} \delta^f_{i,k} w^e_{ij,k}, \quad j = 1,2,\ldots,S. \tag{13}$$

Layer D: Based on Eqs. (13), (7) and (6), error term in this layer is computed as

$$\delta^d_{j,k} = -\frac{\partial J_k}{\partial I^d_{j,k}} = -\frac{\partial J_k}{\partial I^e_{j,k}} \frac{\partial I^e_{j,k}}{\partial O^d_{j,k}} \frac{\partial O^d_{j,k}}{\partial I^d_{j,k}}$$

$$= \delta^e_{j,k} \sum_{\substack{i=1\\ i\neq j}}^{S} O^d_{i,k} \Bigg/ \left(\sum_{i=1}^{s} O^d_{i,k}\right)^2 \qquad j = 1,2,\dots,S. \tag{14}$$

Layer C: Considering the Eqs. (13), (6) and (5), we can also derive the propagating error term of layer C in the same way.

$$\delta^c_{ij,k} = -\frac{\partial J_k}{\partial I^c_{ij,k}} = -\sum_{l=1}^{S} \frac{\partial J_k}{\partial I^d_{l,k}} \frac{\partial I^d_{l,k}}{\partial O^c_{ij,k}} \frac{\partial O^c_{ij,k}}{\partial I^c_{ij,k}}$$

$$= -\sum_{l=1}^{S} \delta^d_{l,k} - e^{\frac{(O^b_{i,k}-C_{ij,k})^2}{\sigma^2_{ij,k}}} T_{ij},$$

$$i = 1,2,\dots,n; \quad j = 1,2,\dots,7, \tag{15}$$

where T_{ij} is an off/on function and can be defined as follows:

$$\begin{cases} T_{ij} = \dfrac{\partial I^d_{l,k}}{\partial O^c_{ij,k}} = 1 & \text{if } O^c_{ij,k} \\ & \quad = \min\{O^c_{1i_1,k},\dots,O^c_{ni_n,k}\}, \\ T_{ij} = 0 & \text{otherwise.} \end{cases} \tag{16}$$

Layer B: Using Eqs. (15), (5) and (4), we have

$$\delta^b_{i,k} = -\frac{\partial J_k}{\partial I^b_{i,k}} = -\sum_{j=1}^{m_j} \frac{\partial J_k}{\partial I^c_{ij,k}} \frac{\partial I^c_{ij,k}}{\partial O^b_{i,k}} \frac{\partial O^b_{i,k}}{\partial I^b_{i,k}}$$

$$= -2\sum_{j=1}^{m_{ij}} \delta^c_{i,k} \frac{(O^b_{i,k} - C_{ij,k})}{\sigma^2_{ij,k}}, \quad i = 1,2,\dots,n. \tag{17}$$

In view of the above definitions, it is easy for us to derive the partial derivative of J_k with respect to four parameters (i.e., $w^a_{i,k}$, $\sigma_{ij,k}$, $C_{ij,k}$ and $w^f_{i,k}$), respectively

$$\frac{\partial J_k}{\partial w^a_{i,k}} = \frac{\partial J_k}{\partial I^b_{i,k}} \frac{\partial I^b_{i,k}}{\partial w^a_{i,k}} = -\delta^b_{i,k} x_{i,k}$$

from Eqs. (17), (4), (18)

$$\frac{\partial J_k}{\partial C_{ij,k}} = \frac{\partial J_k}{\partial I^c_{ij,k}} \frac{\partial I^c_{ij,k}}{\partial C_{ij,k}} = -\delta^c_{ij,k} \frac{2(O^b_{i,k} - C_{ij,k})}{\sigma^2_{ij,k}}$$

from Eqs. (15), (5), (19)

$$\frac{\partial J_k}{\partial \sigma_{ij,k}} = \frac{\partial J_k}{\partial I^c_{ij,k}} \frac{\partial I^c_{ij,k}}{\partial \sigma_{ij,k}} = -\delta^c_{ij,k} \frac{2(O^b_{i,k} - C_{ij,k})^2}{\sigma^3_{ij,k}}$$

from Eqs. (15), (5), (20)

$$\frac{\partial J_k}{\partial w^f_{i,k}} = \frac{\partial J_k}{\partial I^g_{i,k}} \frac{\partial I^g_{i,k}}{\partial w^f_{i,k}} = -\,\delta^g_{i,k} O^f_{i,k}$$

from Eqs. (11), (9). (21)

From Eqs. (18)–(21), the adaptive updated rules for $w^a_{i,k}$, $\sigma_{ij,k}$, $C_{ij,k}$ and $w^f_{i,k}$ can be derived as follows:

$$w^a_{i,k+1} = w^a_{i,k} - \eta_{ai} \frac{\partial J_k}{\partial w^a_{i,k}},$$

$$w^f_{i,k+1} = w^f_{i,k} - \eta_{fi} \frac{\partial J_k}{\partial w^f_{i,k}},$$

$$C_{ij,k+1} = C_{ij,k} - \eta_{cij} \frac{\partial J_k}{\partial C_{ij,k}},$$

$$\sigma_{ij,k+1} = \sigma_{ij,k} - \eta_{\sigma ij} \frac{\partial J_k}{\partial \sigma_{ij,k}}, \tag{22}$$

where $\eta_{ai}, \eta_{cij}, \eta_{\sigma ij}, \eta_{fi}$ denote the learning rates of $w^a_{i,k}, \sigma_{ij,k}, C_{ij,k}, w^f_{i,k}$, respectively.

3.2. Similar Newton (SN) algorithm

In order to use Newton method and release computational burden near the point with extremum, we proposed a similar Newton (SN) algorithm in this subsection. This method is a slight modification of the method reported by [14] and we use it in the above-mentioned FNN. The SN algorithm is employed here to tune the link weights ($w^e_{ij,k}$, $i=1,2,\dots,m$; $j=1,2,\dots,S$) of layer E.

In order to derive the SN learning algorithm simply, we firstly make some changes for Eq. (10).

$$E = \sum_{i=1}^{m} E_{i,r},$$

$$E_{i,r} = \frac{1}{2}\sum_{k=1}^{r} [y_{i,k} - O^g_{i,k}]^2$$

$$= \frac{1}{2}\sum_{k=1}^{r} e^2_{i,k} = E_{i,r-1} + \frac{1}{2} e^2_{i,r}. \tag{23}$$

Based on Newton method, the weights can be tuned as follows [14]:

$$\mathbf{W}^e_{i,r} = \mathbf{W}^e_{i,r-1} - \alpha \mathbf{H}^{-1}_{i,r}\Big|_{\mathbf{W}^e_i=\mathbf{W}^e_{i,r-1}} \frac{\partial E_{i,r}}{\partial \mathbf{W}^e_i}\Big|_{\mathbf{W}^e_i=\mathbf{W}^e_{i,r-1}}, \tag{24}$$

where $\mathbf{W}^e_{i,r} = [w^e_{i1},\dots,w^e_{is}]^{\mathrm{T}}$ represents the weight vector of layer E and the learning rate α is set to be unity by default.

Next, we will use a similar method to derive

$$\frac{\partial E_{i,r}}{\partial \mathbf{W}^e_i}\Big|_{\mathbf{W}^e_i=\mathbf{W}^e_{i,r-1}}, \qquad \mathbf{H}^{-1}_{i,r}\Big|_{\mathbf{W}^e_i=\mathbf{W}^e_{i,r-1}} \quad \text{and} \quad \mathbf{W}^e_{i,r}$$

near the point with extremum.

3.2.1. $\frac{\partial E_{i,r}}{\partial \mathbf{W}^e_i}\Big|_{\mathbf{W}^e_i=\mathbf{W}^e_{i,r-1}}$

Using Eq. (23) and vector $\mathbf{W}^e_{i,r} = [w^e_{i1},\dots,w^e_{is}]^{\mathrm{T}}$, we can get the following equation:

$$\frac{\partial E_{i,r}}{\partial \mathbf{W}^e_i} = \left[\frac{\partial E_{i,r}}{\partial w^e_{i1}},\dots,\frac{\partial E_{i,r}}{\partial w^e_{is}}\right]^{\mathrm{T}} = \sum_{k=1}^{r} e_{i,k}\left[\frac{\partial e_{i,k}}{\partial w^e_{i1}},\dots,\frac{\partial e_{i,k}}{\partial w^e_{is}}\right]^{\mathrm{T}} \tag{25}$$

From Eqs. (23) and (25), we have

$$\frac{\partial E_{i,r}}{\partial \mathbf{W}^e_i}\Big|_{\mathbf{W}^e_i=\mathbf{W}^e_{i,r-1}} = \frac{\partial E_{i,r-1}}{\partial \mathbf{W}^e_i}\Big|_{\mathbf{W}^e_i=\mathbf{W}^e_{i,r-1}} + e_{i,r}\left[\frac{\partial e_{i,r}}{\partial w^e_{i1}},\dots,\frac{\partial e_{i,r}}{\partial w^e_{is}}\right]^{\mathrm{T}}. \tag{26}$$

In [14], the following equation has been proved to be true:

$$\begin{aligned} &\frac{\partial E_{i,r-1}}{\partial \mathbf{W}^e_i}\Big|_{\mathbf{W}^e_i=\mathbf{W}^e_{i,r-1}} = 0 \quad \text{when locating exactly in the point with extremum,} \\ &\frac{\partial E_{i,r-1}}{\partial \mathbf{W}^e_i}\Big|_{\mathbf{W}^e_i=\mathbf{W}^e_{i,r-1}} \approx 0 \quad \text{when near the point with extremum.} \end{aligned} \tag{27}$$

Substituting Eq. (27) into Eq. (26), we can get the following expression (near the point with extremum):

$$\frac{\partial E_{i,r}}{\partial \mathbf{W}^e_i}\Big|_{\mathbf{W}^e_i=\mathbf{W}^e_{i,r-1}} \approx e_{i,r}\left[\frac{\partial e_{i,r}}{\partial w^e_{i1}},\dots,\frac{\partial e_{i,r}}{\partial w^e_{is}}\right]^{\mathrm{T}}, \tag{28}$$

3.2.2. $\mathbf{H}^{-1}_{i,r}|_{\mathbf{W}^e_i=\mathbf{W}^e_{i,r-1}}$.

Referring to the definition of Hesse matrix [14], and with Eq. (23) we have

$$(\mathbf{H}_{i,r})_{pq} = \frac{\partial^2 E_{i,r}}{\partial w^e_{ip}\partial w^e_{iq}} = \sum_{k=1}^{r}\frac{\partial e_{i,k}}{\partial w^e_{ip}}\frac{\partial e_{i,k}}{\partial w^e_{iq}} + \sum_{k=1}^{r} e_{i,k}\frac{\partial^2 e_{i,k}}{\partial w^e_{ip}\partial w^e_{iq}}. \tag{29}$$

Using $e_{i,k} = y_{i,k} - O^g_{i,k}$ and Eqs. (8), (9), we can arrive at the following equation:

$$\frac{\partial e_{i,k}}{\partial w^e_{ip}} = -\frac{\partial O^g_{i,k}}{\partial w^e_{ip}} = -\frac{\partial O^g_{i,k}}{\partial I^g_{i,k}}\frac{\partial I^g_{i,k}}{\partial O^f_{i,k}}\frac{\partial O^f_{i,k}}{\partial w^f_{ip}} = -w^f_{i,k}O^e_{p,k}. \tag{30}$$

Similarly to Eq. (30), we can get

$$\frac{\partial e_{i,k}}{\partial w^e_{iq}} = -w^f_{i,k}O^e_{q,k}. \tag{31}$$

Using Eqs. (30) and (31), we have the following equation:

$$\frac{\partial e_{i,k}}{\partial w^e_{ip}}\frac{\partial e_{i,k}}{\partial w^e_{iq}} = (w^f_{i,k})^2 O^e_{p,k}O^e_{q,k}. \tag{32}$$

Based on Eq. (30), the following expression can be obtained:

$$\frac{\partial^2 e_{i,k}}{\partial w^e_{ip}\partial w^e_{iq}} = -\frac{\partial[w^f_{i,k}O^f_{p,k}]}{\partial w^e_{iq}} = 0. \tag{33}$$

Substituting Eqs. (32), (33) into Eq. (29), we have

$$(\mathbf{H}_{i,r})_{pq} = \sum_{k=1}^{r}\frac{\partial e_{i,k}}{\partial w^e_{ip}}\frac{\partial e_{i,k}}{\partial w^e_{iq}} = \sum_{k=1}^{r}(w^f_{i,k})^2 O^e_{p,k}O^e_{q,k}, \tag{34}$$

$$\mathbf{H}_{i,r} = \sum_{k=1}^{r} (w^f_{i,k})^2 \mathbf{P}^e_k \mathbf{P}^{e\mathrm{T}}_k = \mathbf{H}_{i,r-1} + (w^f_{i,k})^2 \mathbf{P}^e_r \mathbf{P}^{e\mathrm{T}}_r, \tag{35}$$

where $\mathbf{P}^e_r = [O^e_{1,r}, O^e_{2,r}, \ldots, O^e_{s,r}]^{\mathrm{T}}$. Assuming $\mathbf{Q}_{i,r} = \mathbf{H}^{-1}_{i,r}$ and using Eq. (35), we can get

$$\begin{aligned}\mathbf{H}^{-1}_{i,r}\Big|_{\mathbf{W}^e_i=\mathbf{W}^e_{i,r-1}} &= \mathbf{Q}_{i,r}\Big|_{\mathbf{W}^e_i=\mathbf{W}^e_{i,r-1}} \\ &= \mathbf{Q}_{i,r-1} - [\mathbf{P}^{e\mathrm{T}}_r \mathbf{Q}_{i,r-1} \mathbf{P}^e_r]^{-1} \\ &\quad \mathbf{Q}_{i,r-1} \mathbf{P}^e_r \mathbf{P}^{e\mathrm{T}}_r \mathbf{Q}_{i,r-1}.\end{aligned} \tag{36}$$

3.2.3. $\mathbf{W}^e_{i,r}$

Substituting Eq. (30) or Eq. (31) into Eq. (28), we have

$$\frac{\partial E_{i,r}}{\partial \mathbf{W}^e_i}\Big|_{\mathbf{W}^e_i=\mathbf{W}^e_{i,r-1}} \approx -w^f_{i,k} e_{i,r} \mathbf{P}^e_r. \tag{37}$$

Substituting Eqs. (36) and (37) into Eq. (24), we can get the following learning algorithm with guaranteed stability:

$$\begin{aligned}&\mathbf{W}^e_{i,k} \approx \mathbf{W}^e_{i,k-1} + w^f_{i,k} e_{i,k} \mathbf{Q}_{i,k} \mathbf{P}^e_k, \\ &\mathbf{Q}_{i,r} = \mathbf{Q}_{i,r-1} - [\mathbf{P}^{e\mathrm{T}}_r \mathbf{Q}_{i,r-1} \mathbf{P}^e_r]^{-1} \mathbf{Q}_{i,r-1} \mathbf{P}^e_r \mathbf{P}^{e\mathrm{T}}_r \mathbf{Q}_{i,r-1}, \\ &\qquad k = 1, 2, \ldots, r, \\ &\mathbf{Q}_{i,0} = \mathbf{I},\end{aligned} \tag{38}$$

where $\mathbf{Q}_{i,0} \in S \times S$ represents the initial matrix.

3.3. The combining learning (CL) algorithm

The EBP algorithm, which is used very widely in feedforward neural network, is a standard learning algorithm. EBP algorithm converges fast when far away from the point with extremum, while it converges very slowly near the point with extremum where the gradient of error may be almost equivalent to 0 [14].

For the disadvantage of EBP, a similar Newton (SN) algorithm is presented in Section 3.2 and used near the point with extremum.

Therefore, a new learning algorithm that combines the EBP algorithm with the SN algorithm can be established. The basic idea of the coming learning (CL) algorithm is that EBP algorithm is used far away from the point with extremum, while SN algorithm is used near the point extremum.

Next, we define the stopping condition for the proposed EBP and SN learning algorithm.

$$\begin{aligned}&J_k \leqslant \varepsilon_{ebp}, \quad \text{the stopping condition for} \\ &\quad \text{EBP learning algorithm,} \\ &E \leqslant \varepsilon_{sn}, \quad \text{the stopping condition for} \\ &\quad \text{SN learning algorithm,}\end{aligned} \tag{39}$$

where ε_{ebp} is the critical error near the point with extremum. The CL algorithm can be realized as follows:

Step A: Initialize variable:

(a1): Assign initial values to $w^a_{i,0}, \eta_{ai}, w^f_{i,0}, \eta_{fi}, C_{ij,0}, \eta_{cij}, \sigma_{ij,0}, \eta_{\sigma ij}, \mathbf{W}^e_{i,0}$.
(a2): Set values for $\varepsilon_{ebp}, \varepsilon_{sn}$ to give the stopping condition.

Step B: Repeat the following procedure for $k = 1, 2, \ldots, r$:

(b1): Forward calculation: calculate inputs and outputs of neurons for each layer using Eq. (3)–(9), and J_k defined in Eq. (10).
(b2): Back-propagation: calculate the error term to be propagated using Eq. (11)–(17), and the most optimal learning rate using Eq. (52)–(55).
(b3): Weights adjustment: update $w^a_{i,k}, \sigma_{ij,k}, C_{ij,k}, w^f_{i,k}$ using Eq. (22).
(b4): Check whether the stopping condition Eq. (39) is satisfied. If $J_k \leqslant \varepsilon_{ebp}$ is satisfied, go to step C.

Step C: Repeat step D for $i = 1, 2, \ldots, m$.

Step D: Repeat the following procedure for $k = 1, 2, \ldots, r$:

(d1): Calculate the $O^g_{i,k}, e_{i,k}$.
(d2): Update the weight vector $\mathbf{W}^e_{i,k}$ using Eq. (38).

(d3): Check whether the stopping condition is satisfied. If $E \leqslant \varepsilon_{sn}$ is satisfied, stop.

4. Stability analysis

Next, we will develop a convergence theorem to guarantee the stability of the CL learning algorithm used for the above-mentioned FNN.

Theorem 1. *Let η denotes the learning rate for the arbitrary variable w of the above-mentioned FNN. Then the convergence of the algorithm described in Eq.* (22) *is guaranteed if*

$$0 < \eta < \frac{2}{(\partial E_k/\partial w)^2}. \tag{40}$$

Proof. A Lyapunov energy function is defined as follows:

$$V_k = J_k = \tfrac{1}{2}E_k^2. \tag{41}$$

From Eq. (41), we can get

$$\Delta V = V_{k+1} - V_k = \tfrac{1}{2}(E_{k+1}^2 - E_k^2). \tag{42}$$

The error difference, ΔE_k, can be defined as

$$\Delta E_k = E_{k+1} - E_k = \frac{\partial E_k}{\partial w}\Delta w, \tag{43}$$

where $\Delta w = w_{k+1} - w_k = -\eta(\partial J_k/\partial w) = -\eta E_k (\partial E_k/\partial w)$ can be derived from Eq. (22).

Using Eq. (42), we can get

$$\begin{aligned}\Delta V &= \tfrac{1}{2}(E_{k+1} - E_k)(E_{k+1} + E_k)\\ &= \tfrac{1}{2}\Delta E_k(2E_k + \Delta E_k).\end{aligned} \tag{44}$$

Substituting Eq. (43) into Eq. (44), we have

$$\begin{aligned}\Delta V &= \frac{1}{2}\frac{\partial E_k}{\partial w}\eta E_k \frac{\partial E_k}{\partial w}\left(-2E_k + \frac{\partial E_k}{\partial w}\eta E_k \frac{\partial E_k}{\partial w}\right)\\ &= \frac{1}{2}\left(E_k\frac{\partial E_k}{\partial w}\right)^2\left[\left(\frac{\partial E_k}{\partial w}\right)^2\eta^2 - 2\eta\right].\end{aligned} \tag{45}$$

If $\Delta V < 0$, the convergence of the algorithms described in Eq. (22) can be guaranteed. Therefore, we have

$$\left(\frac{\partial E_k}{\partial w}\right)^2\eta^2 - 2\eta < 0. \tag{46}$$

From Eq. (46), we can obtain

$$0 < \eta < \frac{2}{(\partial E_k/\partial w)^2}. \tag{47}$$

Corollary 2. *The most optimal convergence of the learning algorithms described in Eq.* (22) *can be guaranteed if*

$$\eta_{ai}^* = \left[\frac{E_k}{\delta_{i,k}^b x_{i,k}}\right]^2, \quad i = 1,2,\ldots,n,$$

$$\eta_{fi}^* = \left[\frac{E_k}{\delta_{i,k}^g O_{i,k}^f}\right]^2, \quad i = 1,2,\ldots,m,$$

$$\eta_{cij}^* = \frac{1}{4}\frac{E_k^2\sigma_{ij,k}^4}{[\delta_{ij,k}^c]^2[O_{i,k}^b - C_{ij,k}]^2}, \quad i = 1,2,\ldots,n,\ j = 1,2,\ldots,7,$$

$$\eta_{\sigma ij}^* = \frac{1}{4}\frac{E_k^2\sigma_{ij,k}^6}{[\delta_{ij,k}^c]^2[O_{i,k}^b - C_{ij,k}]^4}, \quad i = 1,2,\ldots,n,\ j = 1,2,\ldots,7,$$

where η^* denotes the most optimal learning rate.

Proof. Let

$$f(\eta) = \left[\frac{\partial E_k}{\partial w}\right]^2\eta^2 - 2\eta. \tag{48}$$

Then we can get

$$f'(\eta) = 2\left(\frac{\partial E_k}{\partial w}\right)^2\eta - 2,$$

$$f''(\eta) = 2\left(\frac{\partial E_k}{\partial w}\right)^2 > 0. \tag{49}$$

Let $f'(\eta^*) = 0$, then we have

$$\eta^* = 1\Big/\left[\frac{\partial E_k}{\partial w}\right]^2 = [E_k]^2\Big/\left[\frac{\partial J_k}{\partial w}\right]^2. \tag{50}$$

Seeing from Eq. (49), we can find that $f(\eta)$ has a minimum value at point η^* because $f'(\eta^*) = 0$ and $f''(\eta^*) > 0$. In view of this case, an important conclusion about Eq. (45) can be obtained as follows:

$$\min \Delta V = \Delta V|_{\eta=\eta^*}. \tag{51}$$

Therefore, the most optimal convergence of the learning algorithms described in Eq. (22) can be guaranteed if $\eta = \eta^*$.

Substituting Eq. (18) into Eq. (50), we have

$$\eta^*_{ai} = \frac{E_k^2}{[\partial J_k/\partial w]^2} = \left[\frac{E_k}{\delta^b_{i,k} x_{i,k}}\right]^2, \quad i = 1, 2, \ldots, n. \tag{52}$$

Substituting Eq. (21) into Eq. (50), we can get

$$\eta^*_{fi} = \frac{E_k^2}{[\partial J_k/\partial w]^2} = \left[\frac{E_k}{\delta^g_{i,k} O^f_{i,k}}\right]^2, \quad i = 1, 2, \ldots, m. \tag{53}$$

Substituting Eq. (19) into Eq. (50), we have

$$\eta^*_{cij} = \frac{E_k^2}{[\partial J_k/\partial w]^2} = \frac{1}{4}\frac{E_k^2 \sigma^4_{ij,k}}{[\delta^c_{ij,k}]^2 [O^b_{i,k} - C_{ij,k}]^2}, \quad i = 1, 2, \ldots, n, \; j = 1, 2, \ldots, 7. \tag{54}$$

Substituting Eq. (20) into Eq. (50), we have

$$\eta^*_{\sigma ij} = \frac{E_k^2}{[\partial J_k/\partial w]^2} = \frac{1}{4}\frac{E_k^2 \sigma^6_{ij,k}}{[\delta^c_{ij,k}]^2 [O^b_{i,k} - C_{ij,k}]^4}, \quad i = 1, 2, \ldots, n, \; j = 1, 2, \ldots, 7. \tag{55}$$

5. Manufacturing process control

Increasing the productivity of manufacturing process is a principal concern for CNC machine tools. Nevertheless, a common drawback of CNC machine tools is that the part-programmer must prescribe conservative cutting parameters to avoid tool breakage and excessive tool wear [33,34]. These conservative cutting parameters will result in the machining time increasing, so a new FNN controller for CNC machine tools is required. The FNN controller should automatically regulate the cutting parameters to maintain a constant level of cutting force, so that the CNC machine tools can maintain the maximum working ability during machining.

In this section, we firstly describe how the above-mentioned FNN acts as the intelligent controller for the CNC machine tools, and then give the experiment results in details.

5.1. Fuzzy-neural network control system and experimental setup

Based on the above-mentioned FNN, an intelligent controller is proposed to maintain a constant force under varying cutting conditions with adjustable feed rates.

Fig. 2 shows the block diagram of the intelligent control system in milling operation. In Fig. 2, the signals that are fed into the FNN can be calculated as follows:

$$e_k = F_r - F_{m,k},$$
$$ec_k = F_{m,k} - F_{m,k-1}, \tag{56}$$

where F_r and $F_{m,k}$ denote, respectively, the reference cutting force and the measured cutting force.

During the on-line learning process, the fuzzy control rules stated in (Eq. (1)) and the input command of the plant (V_{com}) have been tuned to get constant cutting force. From Fig. 2, we have the following equation:

$$V_{\text{com}} = V_k + \Delta V_k. \tag{57}$$

The configuration of the experimental setup is shown in Fig. 3. The machine tool used in the experiment is the commercial vertical-machining center equipped with the FANUC CNC 15M. The cutting force signal was obtained by using the Amplifier (YE5850) and table-type dynamometer (Kistler 9257A), and then recorded on a Pentium-PC through a data acquisition board (PCL818HG). The above-mentioned CL learning algorithm is employed to tune the feed rate command (V_{com}). As a result, V_{com} is fed back into the FANUC CNC system to maintain a constant level of cutting force.

5.2. Fuzzy sets and fuzzy control rules

Described as Section 2, seven fuzzy sets (labeled as: NB, NM, NS, ZO, PS, PM and PB) are used for fuzzy control rules and the membership functions for these seven fuzzy sets are all in bell-shaped. Here, the seven fuzzy sets for the input and output linguistic variables of the FNN control system have both been designed in the same range of [−6, +6] (Fig. 4).

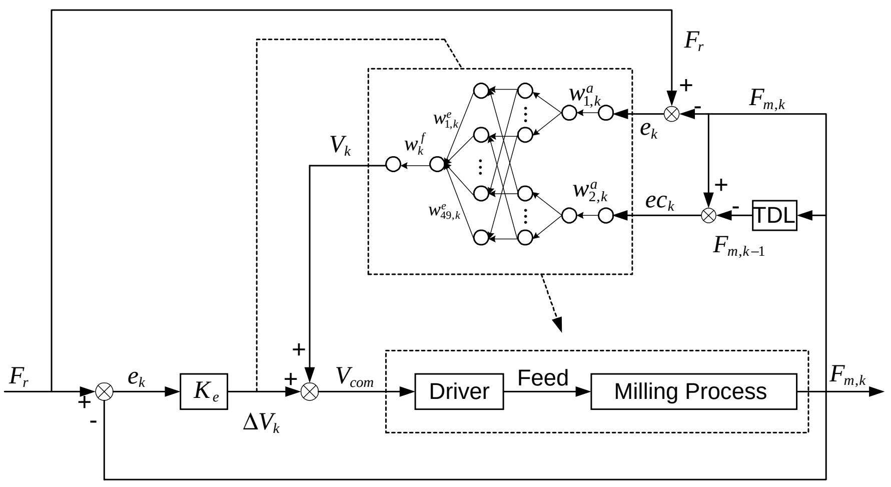

Fig. 2. Block diagram of the fuzzy neural network control system in milling.

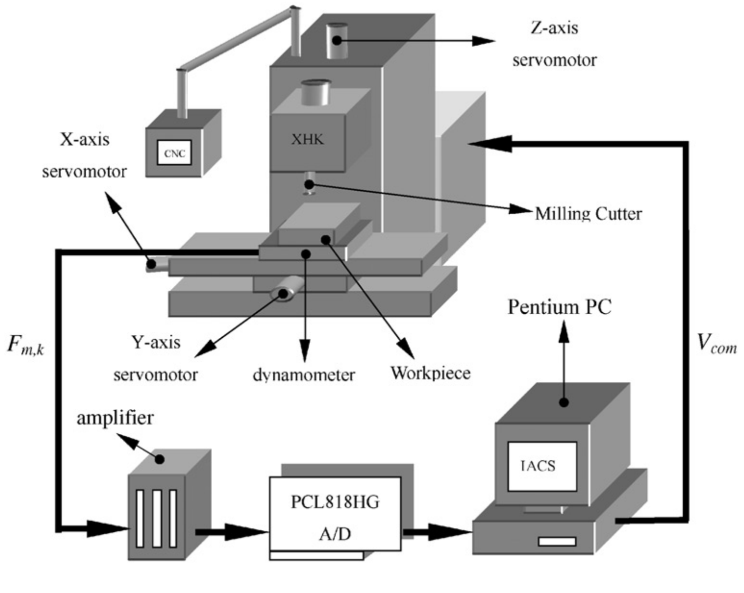

Fig. 3. Experimental setup.

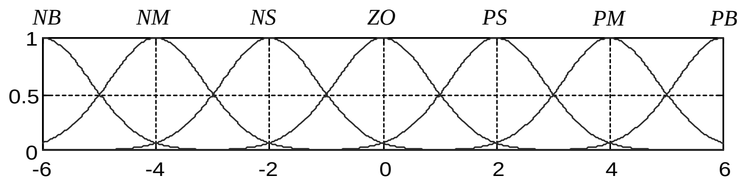

Fig. 4. Seven fuzzy sets for the input and output linguistic variables of the FNN control system.

Table 1
Fuzzy control rules for manufacturing process

	V/ec						
	NB	NM	NS	ZO	PS	PM	PB
NB	PB	PB	PB	PB	PM	PS	ZO
NM	PB	PB	PM	PM	PS	ZO	NS
NS	PB	PM	PM	PS	ZO	NS	NM
ZO	PM	PS	PS	ZO	NS	NS	NM
PS	PS	PS	ZO	NS	NM	NM	NB
PM	PS	ZO	NS	NM	NM	NB	NB
PB	ZO	NS	NM	NB	NB	NB	NB

Table 2
Detailed milling conditions

Machine center	XHK
CNC system	Fanuc CNC 15M
Workpiece material	Ti_6Al_4V
Milling cutter	Ball-end milling cutter
Spindle speed	1200 r/min
Table dynamometer	Kister 9257A

5.3. First experimental results

Once the shape of the fuzzy sets is given by Fig. 4, The 49 fuzzy control rules are formulated based on the knowledge of well-experienced manufacturing engineers (Table 1). With the defuzzification method of singleton, the 49 fuzzy control rules are assigned as the initial values of connection weights, $\mathbf{W}^e_{i,0}$

Sets of cutting tests have been performed to verify the performance of the above-mentioned FNN control system. Below, a typical milling experiment on a titanium alloy Ti_6Al_4V is given (Table 2 and Fig. 5). Shown as Fig. 5, the axial depths of cut vary during the manufacturing process. Therefore, the control system should aim to get constant cutting force with the varying axial depths of cut.

Fig. 6(a) shows the feed rate of ball-end milling cutter with the varying axial depths of cut. Seeing from Fig. 6(a) and (b), we can find that the feed rate varies with the varying axial depth and the cutting force can keep constant (the cutting force = 750 N).

Fig. 7 shows the on-line learning procedure of CL algorithm. From Fig. 7, we can conclude:

(1) The minimum feedback error and the feedback error gain are, respectively, set as $E = \varepsilon_{sn} = 0.0001$ and $K_e = 0.015$.

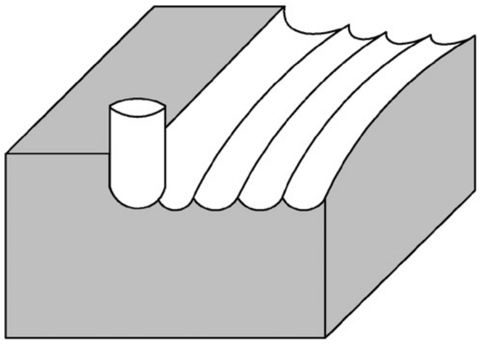
Fig. 5. Cutting geometry.

(2) When the cutter starts to engage the workpiece, the EBP algorithm is used. At this time, the EBP algorithm converges very fast. However, it converges very slowly when the error signal reaches $E = \varepsilon_{ebp} = 0.01$ ($t_1 = 70$ ms).

(3) At this time, the SN algorithm is used. It is seen that the SN algorithm converges very fast and the error signal reaches $E = \varepsilon_{sn} = 0.0001$ very quickly ($t_2 = 90$ ms).

(4) It is satisfactory in manufacturing process that the total learning time of CL algorithm is 90 ms before the error signal reaches $E = \varepsilon_{sn} = 0.0001$.

Before $t = 90$ ms, we get the training data from table dynamometer to tune the above-mentioned FNN. After tuning, we can get the tuned fuzzy sets (or membership functions) of input linguistic variables (ec and e) and fuzzy control rules, $\mathbf{W}^e_{i,k}$.

Fig. 8 (a) and (b) shows, respectively, the tuned seven fuzzy sets of e and ec. The tuned fuzzy control rules ($\mathbf{W}^e_{i,k}$) are shown in Table 3. From Fig. 8 and Table 3, we can find that fuzzy sets and fuzzy control rules have been adaptive adjusted after the learning process.

5.4. Second experimental results

5.5. Third experimental results

In order to test the effect/form of the trained fuzzy partitioning strategy, the fuzzy control rules (Table 3) were used as the initial condition of the second experiment. In the same way, the trained fuzzy control rules in the second experiment were also used as the initial condition of the third experiment.

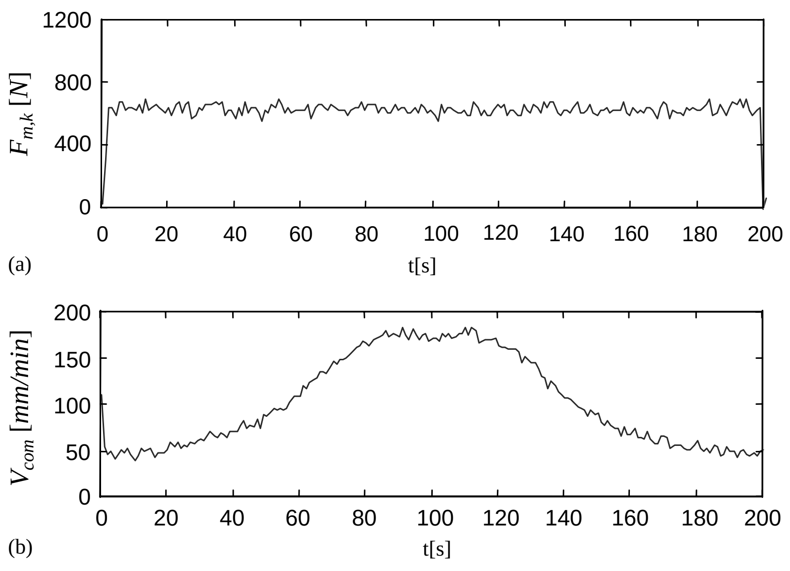

Fig. 6. (b) Feed rate.

From Figs. 7, 10 and 13, we can find that the trained fuzzy partitioning strategy can accelerate the learning process ($t \neq 90 \Rightarrow 60 \Rightarrow 40$ ms). The average overshot of Figs. 6, 11 and 14 are, respectively, 16.7%, 11.5% and 8.3%. Therefore, we can also find that the trained fuzzy partitioning strategy can reduce the average overshot.

5.6. *Comparison with other methods*

Compared with conventional self-tuning PID controller, the above-mentioned FNN is robust. The reasons are that the CL algorithm often begins with the fuzzy control rules and the trained fuzzy partitioning strategy can reduce the average overshot.

Chen et al. [13] proposed a feedforward multilayer connectionist network to realize the fuzzification, fuzzy operator and defuzzification, but the scaling factors $(w^a_{i,k}, w^f_{i,k})$ cannot be tuned. In our FNN, we can tune the scaling factors $(w^a_{i,k}, w^f_{i,k})$.

Wang et al. [35] developed a class of adaptive FNN to on-line tune the shape of membership functions. Unfortunately, the tuning rules are pre-designed based on the knowledge of well-experienced manufacturing engineers and cannot be adjusted according to the manufacturing process. Our tuning method proposed in this paper can be adjusted by on-line learning.

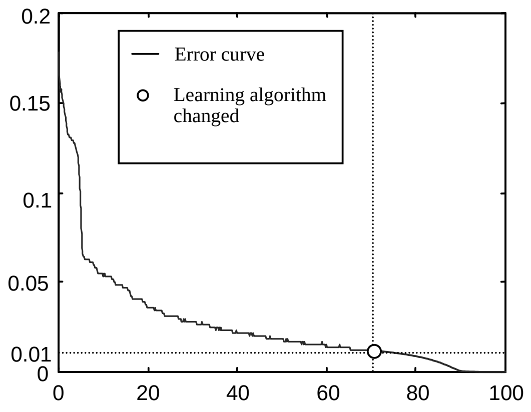

Fig. 7. Learning procedure of CL algorithm.

Lin et al. [24] proposed an effective learning algorithm based on EBP for FNN. However, this paper cannot solve two basic problems: (1) EBP algorithm converges very slowly near the point with extremum; (2) The stability of learning algorithm cannot be guaranteed. We have successfully solved these two difficult problems with the above-mentioned CL learning algorithm, Theorem 1 and Corollary 1.

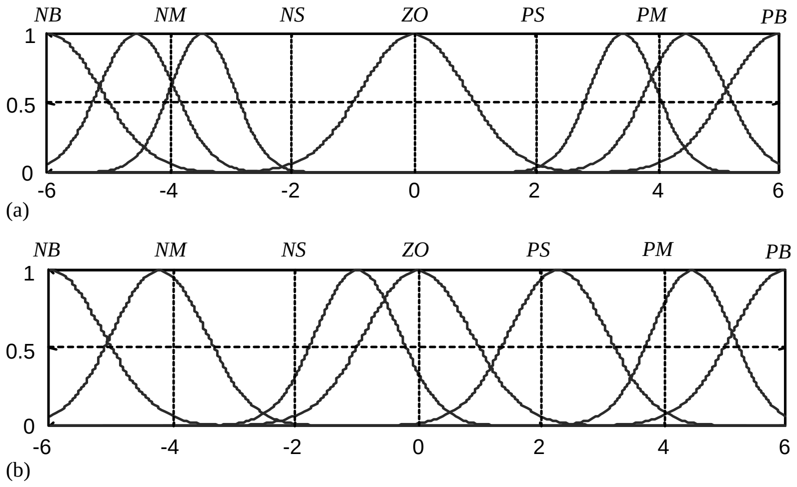

Fig. 8. Seven fuzzy sets for input linguistic variable (a) (*ec*) and (b) (*e*) after tuning.

Table 3
Fuzzy control rules for manufacturing process

e/*V*/*ec*	NB	NM	NS	ZO	PS	PM	PB
NB	PB	PB	PM*	PB	PB*	PS*	ZO
NM	PB	PM*	PM	PM	PS	PS	ZO*
NS	PB	PM	PM	PM*	ZO	NS	NS
ZO	PB*	PS	PS	ZO	NS	NS	NM
PS	PM*	ZO*	ZO	NM*	NM	NM	NM*
PM	PS	ZO	NM*	NM	NB*	NM*	NB
PB	ZO	NS	NM	NB	NB	NB	NB

*: Fuzzy control rules after tuning.

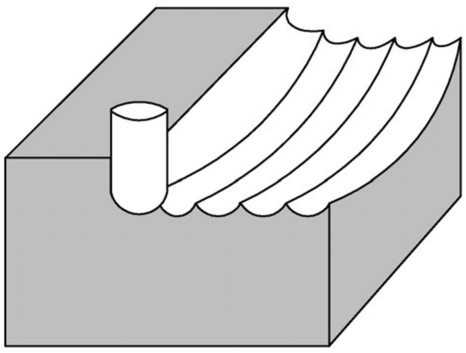

Fig. 9. Cutting geometry.

Tarng et al. [33,34] proposed a new adaptive control system based on neural network and fuzzy logic to maintain the constant cutting force. However, the

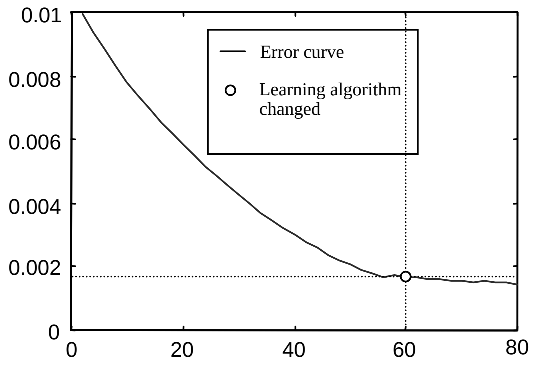

Fig. 10. Learning procedure of CL algorithm.

overshot of cutting force is larger than that of our FNN control system during the full immersion of cut.

6. Conclusions

In this paper we proposed a new FNN which can be applied to manufacturing process control. The proposed FNN has the following attractive features:

(1) It can automatically realize fuzzification, fuzzy inference, defuzzification.

(2) The parameters of FNN, such as scaling factors, membership functions and fuzzy control rules, etc. can be dynamically tuned by learning.

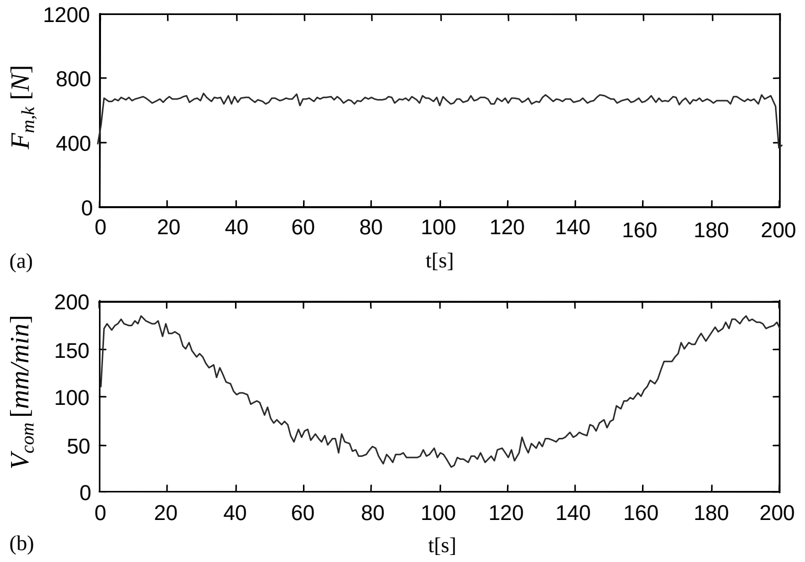

Fig. 11. (a) Cutting force; (b) feed rate.

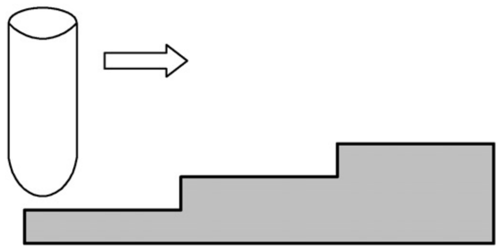

Fig. 12. Cutting geometry.

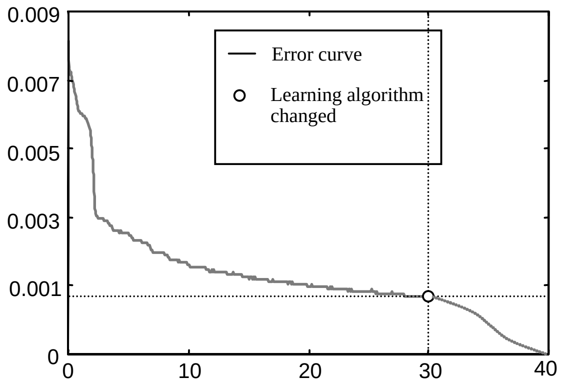

Fig. 13. Learning procedure of CL algorithm.

(3) The CL learning algorithm used in FNN is divided into two phases. The first one is an error back-propagation (EBP) training phase, and the second one is a similar Newton (SN) training phase. The SN learning algorithm has lighter computational burden than Newton method.

(4) The CL learning algorithm makes it possible to escape from the point with extremum.

(5) The theorem about the most optimal learning rate based on Lyapunov's direct method can guarantee the stability of manufacturing process control system.

(6) The real-time control program can be given easily with the BOLAND C++ language.

From the above-mentioned experimental results, we can find that the proposed FNN and CL learning algorithm are applicable to the manufacturing process control. In the near future, we will focus on the following research work:

(1) *Stopping condition*: Selecting the stopping condition ε_{ebp} for EBP learning algorithm is subjective. How to determine the stopping condition automatically is an important problem.
(2) *Rule elimination*: A drawback for FNN is that too many rules are needed. It is necessary for us to provide an algorithm to perform rule combination like [4,31].

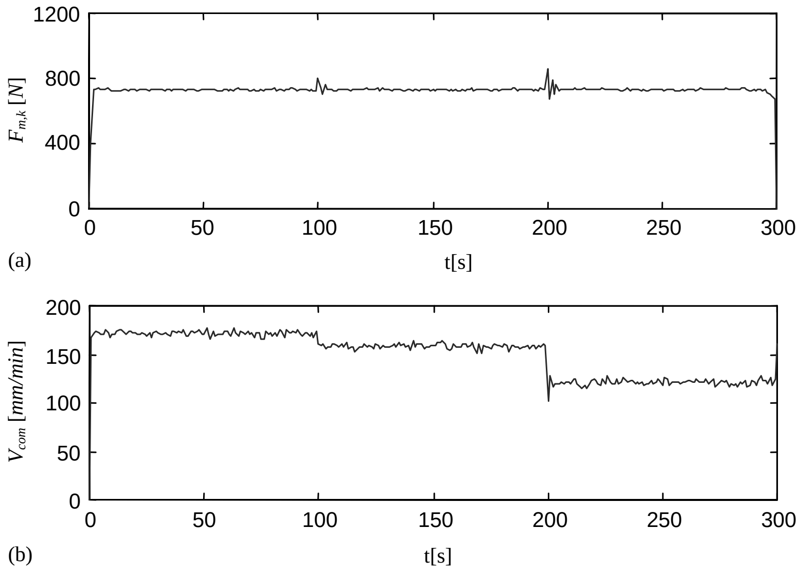

Fig. 14. (a) Cutting force; (b) feed rate.

References

[1] K. Asakawa, H. Takagi, Neural networks in Japan, Commun. ACM 37 (1994) 106–112.

[2] J.F. Bernard, Use of rule-based system for process control, IEEE Contr. System Mag. 8 (1988) 3–13.

[3] M. Brown, C. Harris, Neurofuzzy Adaptive Modelling and Control, Prentice-Hall, Englewood Cliffs, NJ, 1994.

[4] C.T. Chao, C.C. Teng, Implementation of a fuzzy inference system using a normalized fuzzy neural network, Fuzzy Sets and Systems 75 (1995) 17–31.

[5] J.S. Chai, S.H. Tan, Q. Chen, A general fuzzy control scheme for nonlinear processes with stability analysis, Fuzzy Sets and Systems 100 (1998) 179–195.

[6] C.M. Cheng, N.W. Rees, Stability analysis of fuzzy multivariable systems: vector Lyapunov function approach, IEE Proc. Control Theory Appl. 144 (1997) 403–412.

[7] Y.C. Chen, C.C. Teng, A model reference control structure using a fuzzy neural network, Fuzzy Sets and Systems 73 (1995) 291–312.

[8] R.X. Du, M.A. Elbestrawi, S. Li, Tool condition monitoring in turning using fuzzy set theory, Int. J. Mach. Tool Manufact. 32 (1992) 781–796.

[9] G. Feng, S.G. Cao, C.K. Chak, Design of fuzzy control systems with guaranteed stability, Fuzzy Sets and Sytems 85 (1997) 1–10.

[10] T. Hasegawa, S. Horikawa, T. Furuhashi, On design of adaptive fuzzy controller using fuzzy neural networks and a description of its dynamical behavior, Fuzzy Sets and Systems 71 (1995) 3–23.

[11] S.J. Horikawa, T. Furuhashi, Y. Uchikawa, On fuzzy modeling using fuzzy neural networks with the back-propagation algorithms, IEEE Trans. Neural Networks 3 (1992) 801–806.

[12] G.C. Hwang, S.C. Lin, A stability approach to fuzzy control design for nonlinear systems, Fuzzy Sets and Systems 48 (1992) 279–287.

[13] J.S. Jang, Self-learning fuzzy controllers based on temporal back propagation, IEEE Trans. Neural Networks 3 (1992) 723–741.

[14] N.B. Karayiannis, A.N. Venetsanopoulos, Artificial Neural Networks, Learning Algorithm, Performance Evaluation and Applications, Kluwer Academic Publishers, Dordrecht, 1993.

[15] W.C. Kim, S.C. Ahn, W.H. Kwon, Stability analysis and stabilization of fuzzy state space models, Fuzzy Sets and Systems 71 (1995) 131–142.

[16] J.B. Kiszka, M.M. Gupta, P.N. Nikiforuk, Energetistic stability of fuzzy dynamic systems, IEEE Trans. Systems, Man Cybern. SMC-15 (1985) 783–791.

[17] R. Krishnapuram, J. Lee, Fuzzy-set-based hierarchical networks for information fusion in computer vision, Neural Networks 5 (1992) 335–350.

[18] C.C. Ku, K.Y. Lee, Diagonal recurrent neural networks for dynamic systems control, IEEE Trans. Neural Network 6 (1995) 144–155.

[19] S.R. Kumar, D.D. Majumder, Application of FCSs to industrial processes, Automatica 13 (1997) 235–242.

[20] R.J. Kuo, Multi-sensor integration for intelligent control of machining through artificial neural networks and fuzzy modeling, Ph.D. Dissertation, The Pennsylvania State University, 1994.

[21] R.J. Kuo, P.H. Cohen, Manufacturing process control through integration of neural networks and fuzzy model, Fuzzy Sets and Systems 98 (1998) 15–31.

[22] K.M. Lee, D.H. Kwak, H. Leekwang, Tuning of fuzzy models by fuzzy neural networks, Fuzzy Sets and Systems 76 (1995) 47–61.

[23] R.D. Lenone, R. Capparuccia, E. Merelli, A successive overrelaxation backpropagation algorithm for neural-network training, IEEE Trans. Neural Network 9 (1998) 381–388.

[24] C.T. Lin, C.J. Lin, C.S.G. Lee, Fuzzy adaptive learning control network with on-line neural learning, Fuzzy Sets and Systems 71 (1995) 25–45.

[25] Z.A. Luo, P. Tseng, Analysis of an approximate gradient projection method with application to the backpropagation algorithm, Optim. Methods Software 4 (1994) 85–102.

[26] L.J. Maguire, T.M. MeGinnity, L.J. McDaid, A fuzzy neural network for approximate fuzzy reasoning, in: Hybrid Intelligent Systems: Fuzzy Logic, Neural Networks and Genetic Algorithm Da Rnan (Ed.), Kluwer Academic Publisher, Dordrecht, 1997, pp. 35–58 (Chapters 8 and 9).

[27] S. McLoone, M.D. Brown, G. Irwin, A hybrid linear/nonlinear training algorithm for feedforward neural networks, IEEE Trans. Neural Network 9 (1998) 669–684.

[28] K. Michels, Numerical stability analysis for a fuzzy or neural network controller, Fuzzy Sets and Systems 89 (1997) 335–350.

[29] Minyou Chen, D.A. Linkens, A hybrid neuro-fuzzy PID controller, Fuzzy Sets and Systems 99 (1998) 27–36.

[30] M.M. Polycarpou, P.A. Ioannou, Learning and convergence analysis of neural-type structured networks, IEEE Trans. on Neural Networks 3 (1992) 39–50.

[31] J.J. Shann, H.C. Fu, A fuzzy neural network for rule acquiring on fuzzy control systems, Fuzzy Sets and Systems 71 (1995) 345–357.

[32] K. Tanaka, M. Sugeno, Stability analysis and design of fuzzy control systems, Fuzzy Sets and Systems 45 (1992) 135–156.

[33] Y.S. Tarng, S.T. Hwang, A neural network controller for constant turning force, Int. J. Mach. Tools Manufact. 4 (1994) 453–460.

[34] Y.S. Tarng, S.T. Cheng, Fuzzy control of feed rate in end milling operations, Int. J. Mach. Tools Manufact. 4 (1993) 643–650.

[35] L.X. Wang, Adaptive Fuzzy Systems and Control, Prentice-Hall, Englewood Cliffs, NJ, 1994.

[36] L.X. Wang, Combining mathematical model and heuristics into controllers: an adaptive fuzzy control approach, Fuzzy Sets and Systems 89 (1997) 151–156.

[37] L.X. Wang, J.M. Mendel, Back-propagation fuzzy system as nonlinear dynamic system identifiers, IEEE International Conference on Fuzzy system, 1992, pp. 1409–1418.

[38] S. Yamazkai, S. Miyamoto, H. Ihara, Fuzzy control for automatic train operation system, Proceedings of the Fourth IFAC/IFIP/IFORS International Congress on control in Transportation Systems, Baden, 1983.

[39] S. Yasunobu, S. Miyamoto, Automatic train operation by predictive fuzzy control, Ind. Appl. Fuzzy Control (1985) 835–838.

[40] H. Ying, Practical design of nonlinear fuzzy controllers with stability analysis for regulating processes with unknown mathematical models, Automatica 30 (1994) 1185–1195.

[41] Y.Q. Zhang, A. Kandel, Compensatory neurofuzzy systems with fast learning algorithms, IEEE Trans. Neural Networks 9 (1998) 83–105.