On the Equivalence of a Table Lookup (TL) Technique and Fuzzy Neural Network (FNN) With Block Pulse Membership Functions (BPMFs) and Its Application to Water Injection Control of an Automobile

Chi-Hsu Wang, *Fellow, IEEE*, and Jung-Sheng Wen, *Member, IEEE*

***Abstract*—This paper presents an alternative method to design a fuzzy neural network (FNN) using a set of nonoverlapped block pulse membership functions (BMPFs), and this FNN with nonoverlapped BPMFs will be shown to be equivalent to the conventional table lookup (TL) technique. Therefore, the hidden links between TL and FNN techniques are revealed in this paper that provides a methodology to design a TL controller based on the FNN design concept. In order to do so, a new direct formula is first developed to generate the fuzzy rules from the premise part in FNN. This direct formula not only guarantees a one-to-one mapping that maps the fuzzy membership functions onto the fuzzy rules, but also alleviates the coding effort during hardware implementation. It is further elaborated that the FNN with nonoverlapped BPMFs has the advantage of faster online training that requires less computation time, but at the cost of more memory requirement to store the fuzzy rules. The application of this new approach has been applied successfully in the water injection control of a turbo-charged automobile with excellent results.**

***Index Terms*—Fuzzy neural network (FNN), membership functions (MFs), optimal training, table lookup (TL) controller.**

I. Introduction

THE FUZZY neural network (FNN) has been shown to have tremendous impact on engineering applications in the last decade [1]–[4]. Several kinds of FNNs have also been developed for different kinds of applications, such as real-time intelligent adaptive control [5], [6], image processing [7], [8], etc. One of the advantages of FNNs is that it can be trained to suit the real physical environment with either offline or online trainings [9], [10]. Furthermore, the table lookup (TL) technique has been applied extensively for hardware implementation of engineering applications during the last decade [11]–[16]. However, the TL technique is a direct mapping approach without any embedded training strategy. Depending on the requirements of different applications, the FNN and TL techniques can both be applied successfully in engineering applications. To be more specific, fuzzy lookup tables were proposed in [11] to speed up the online learning fuzzy controller with a PD controller. However, the relationships between TL and FNN have never been discussed in a formal and systematic way. Therefore, the major purpose of this paper is to reveal the hidden links between TL and FNN techniques in a rigorous manner. For this to happen, a new direct formula will be first proposed to generate the fuzzy rules in FNN. Then, a special kind of FNN with block pulse membership functions (BPMFs) will be defined. Finally, the new direct formula will be adopted to show the equivalence of the FNN with BPMFs and TL techniques.

It is well known that the fuzzy rules in an FNN must be generated from all the possible combinations of the membership functions (MFs) for all fuzzy input variables, which is very hard to explain using equations. This has lead to the confusing expressions in the research articles [1]–[4] about the generation of fuzzy rules in FNN. Thus, there have been no common guidelines to map the fuzzy MFs onto the fuzzy rules during the implementation of an FNN. Nested looping is usually adopted for the generation of fuzzy rules in coding without a unified ordinal sequence. To alleviate the earlier disadvantages, a new direct formula will be proposed in this paper to generate the fuzzy rules without nested looping and yet guarantee a one-to-one mapping that maps the fuzzy MFs onto the fuzzy rules. This will not only remove the confusing part in the generation of fuzzy rules in FNN, but also the implementation effort of FNN is alleviated. This direct fuzzy rule generation will also provide a unified ordinal sequence for fuzzy rules in FNN.

Further, the conventional MFs adopted in an FNN are always jointly overlapped with each other. The advantage for this overlapped arrangement is that the number of fuzzy rules can be reduced to a minimum. However, it will take more effort to train the weighting factors in either online or offline applications. In [17], the block pulse functions were proposed to approximate time functions to estimate the parameters of a permanent magnetic dc motor, and this will ease the online fault detection using the FNN. In this paper, an alternative architecture of an FNN with BPMFs will be presented. The BPMFs are

Manuscript received April 20, 2007; revised September 20, 2007 and December 7, 2007. This work was supported by the Program for Promoting Academic Excellence of Universities (Phase II) under Grant NSC96-2752-E-027-001-PAE. This paper was recommended by Associate Editor X. Tuan.

C.-H. Wang is with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan, R.O.C. (e-mail: cwang@cn.nctu.edu.tw).

J.-S. Wen was with the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan, R.O.C. He is now with the Department of Computer and Communication Engineering, Technology and Science Institute of Northern Taiwan, Taipei 112, Taiwan, R.O.C. (e-mail: wen@tsint.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.org.

Digital Object Identifier 10.1109/TSMCC.2008.923869

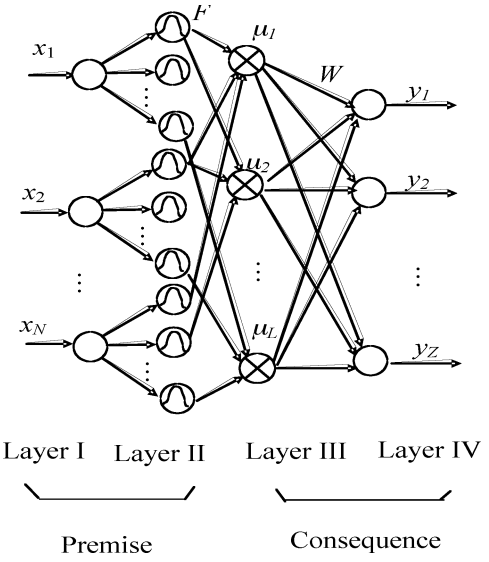

Fig. 1. FNN architecture.

disjoint with each other and cascaded successively. The FNN with BPMFs in this paper can be shown to be equivalent to the conventional TL technique for hardware implementation via the direct new formula for generating fuzzy rules.

It can also be shown that the FNN with BPMFs (or the TL technique) will need more memory to achieve better results by having more nonoverlapped intervals for input fuzzy variables. However, the update of the weighting factor is very simple due to the fact that it is independent of the neighboring variables. On the other hand, the conventional FNN with few overlapped MFs for each input fuzzy variables will need less memory to store the fuzzy rules. But the update of weighting factors will require more computing power with a special algorithm to achieve training convergence. This may not be feasible for online applications. The water injection control for a turbo-charged automobile is illustrated in this paper as the application of the proposed techniques.

II. FNN With Direct Fuzzy Rule Generation

Fig. 1 shows the FNN architecture [1], [2] adopted in this paper. Layer 1 consists of input fuzzy variables, layer II consists of fuzzy MFs for input fuzzy variables, layer III is for the implied fuzzy rules by considering all the cases of the MFs, and layer IV is the output part.

Between layer III and layer IV, there is a simple two-layer neural network with weighting factors, and the weighting factors will be tuned by a training process to have the desired output in layer IV. However, the generation of fuzzy rules was not explicitly discussed in previous research literature [3], [4]. For instance, the specific fuzzy rule l was not defined in [3], and [4], and the only way to generate the fuzzy rules is via nested looping without a unified ordinal sequence. The execution of nested looping in FNN coding is also not efficient, especially in a real-time environment. This has imposed the burden on software implementation of the FNN. Therefore, the following context will be focused on the development of a systematic and unified way to generate the fuzzy rules without nested looping. The fuzzy rules in layer III can be inferred as

$$\text{Rule l:}\quad \text{If } x_1 \text{ is } F_1^{r_1} \text{ and } \cdots \text{ and } x_N \text{ is } F_N^{r_N} \text{ then } y_1 \text{ is } w_1^l \text{ and } \cdots \text{ and } y_Z \text{ is } w_Z^l. \quad (1)$$

Since every BPMF is disjoint with each other, only a single rule is activated each time. Rule l corresponding to the activated rule can be obtained from the following equation:

$$\begin{aligned} l &= r_1 + R_1(r_2) + R_1R_2(r_3) + \cdots + R_1R_2\cdots R_{N-1}(r_N) \\ &= r_1 + \sum_{j=2}^{N}\left(\prod_{i=1}^{j-1} R_i\right) r_j \end{aligned} \quad (2)$$

where R_i is the number of MFs for fuzzy variable x_i ($i = 1, 2, \ldots, N$), $F_i^{r_i}$ is the r_ith MF for fuzzy variable x_i in Rule l, and w_j^l is the MF for y_j ($j = 1, 2, \ldots, Z$) in Rule l. It is also obvious that r_i is an integer number with $r_i \geq 0$. Theorem 1 shows the fact that (2) can produce all the $\boldsymbol{l}$'s in the range of $0 \leq l \leq (\prod_{i=1}^{N} R_i) - 1$ with a one-to-one mapping from MFs to fuzzy rules.

Theorem 1: Let $A = \{(r_1, r_2, \ldots, r_N) | r_i \in I, 0 \leq r_i \leq R_i - 1, i = 1, 2, \ldots, N\}$, $B = \{l | l \text{ is generated by (2)}\}$.

Then, (2) will generate a one-to-one mapping that maps A onto B.

Proof: We need to show the validities of necessary and sufficient conditions

Sufficient condition: For any pair $(r_1, r_2, \ldots, r_N)$ in A, there is a unique l in B.

If $r_1 = 0,\ r_2 = 0, \ldots, r_N = 0,$ then

$$l = 0 + R_1(0) + R_1R_2(0) + \cdots + R_1R_2\cdots R_{N-1}(0) = 0$$

which is the minimum number of $\boldsymbol{l}$.

If $r_1 = R_1 - 1,\ r_2 = R_2 - 1, \ldots, r_N = R_N - 1$, then

$$\begin{aligned} l &= (R_1 - 1) + R_1(R_2 - 1) + R_1R_2(R_3 - 1) \\ &\quad + \cdots + R_1R_2\cdots R_{N-1}(R_N - 1) \\ &= R_1 + R_1(R_2 - 1) + R_1R_2(R_3 - 1) \\ &\quad + \cdots + R_1R_2\cdots R_{N-1}(R_N - 1) - 1 \\ &= R_1(1 + (R_2 - 1) + R_2(R_3 - 1) \\ &\quad + \cdots + R_2\cdots R_{N-1}(R_N - 1)) - 1 \\ &= R_1(R_2 + R_2(R_3 - 1) + \cdots \leq \\ &\quad + R_2\cdots R_{N-1}(R_N - 1)) - 1 \\ &= R_1R_2(1 + (R_3 - 1) \\ &\quad + \cdots + R_3\cdots R_{N-1}(R_N - 1)) - 1 \\ &\cdots \\ &= \left(\prod_{i=1}^{N} R_i\right) - 1 = L \end{aligned}$$

which is the maximum number of $\boldsymbol{l}$.

It is obvious that, if $\{0 < r_i < R_i - 1,\ i = 1, \ldots, N\}$, then

$$1 < l < \left(\prod_{i=1}^{N} R_i\right) - 1.$$

Necessary condition: For any $\boldsymbol{l}$ in the range of $0 \le l \le (\prod_{i=1}^{N} R_i) - 1$, we can find a set of $\{r_1, r_2, \ldots, r_N, 0 \le r_i < R_i, i = 1, \ldots, N\}$ by using the following algorithm.

If $0 \le l < R_1$, then $r_1 = l, r_2 = r_3 = \cdots = r_N = 0$.
If $R_1 \le l < R_1 R_2$, then
$r_1 = l(\text{mod})R_1, r_2 = l\backslash R_1, r_3 = r_4 = \cdots = r_N = 0$.
If $R_1 R_2 \le l < R_1 R_2 R_3$, then

$$\begin{cases} r_1 = l(\text{mod})R_1, r_2 = (l\backslash R_1)(\text{mod})R_2, r_3 = l\backslash(R_1 R2) \\ r_4 = r_5 = \cdots = r_N = 0 \end{cases}$$

. . . . . .
. . . . . .
If $R_1 R_2 \cdots R_{N-1} \le l < R_1 R_2 \cdots R_N$, then

$$\begin{cases} r_1 = l(\text{mod})R_1, r_2 = (l\backslash R_1)(\text{mod})R_2 \\ r_3 = (l\backslash(R_1 R_2))(\text{mod})R_3, \ldots, r_N = l\backslash(R_1 R_2 \cdots R_{N-1}) \end{cases}$$

where

A (mod) B = the remainder of the division of A over B;

$A\backslash B$ = the quotient of the division of A over B.

The main theme of this algorithm is based on (2) to reversely find all the r_i's for a specific $\boldsymbol{l}$. Q.E.D.

The final outcome for each output $\boldsymbol{y}_j$ can be deducted as

$$y_j = \sum_{i=1}^{L} w_j^i \mu_i \Big/ \sum_{i=1}^{L} \mu_i \ (j = 1,\ 2, \ldots,\ Z) \tag{3}$$

where

$$\mu_l = \prod_{i=1}^{N} F_i^{r_i} (\boldsymbol{l} = 0, 1, 2, \ldots, L)\ (r_i = 0, 1, 2, \ldots, R_i - 1). \tag{4}$$

Example 1: For three fuzzy variables $\{\boldsymbol{x}_1,\ \boldsymbol{x}_2,\ \boldsymbol{x}_3\}$, assume $R_1 = 2,\ R_2 = 2,\ R_3 = 2$, generate all fuzzy rules in Fig. 1 by (2).

Solution: There are $R_1 R_2 R_3 = 8$ fuzzy rules, i.e., $\{\mu_1,\ \mu_2,\ \mu_3,\ \mu_4,\ \mu_5,\ \mu_6,\ \mu_7,\ \mu_8\}$. From (2), we have

$$l = r_1 + R_1\,(r_2) + R_1 R_2\,(r_3) = r_1 + 2\,(r_2) + 4\,(r_3)\,.$$

Table I summarizes the result.

Furthermore, for any l in the range of $0 < l < (\prod_{i=1}^{N} R_i) - 1$, a unique set $\{r_1,\ r_2, \ldots,\ r_N\}$ can be found by Theorem 1.

Example 2: For three fuzzy variables $\{\boldsymbol{x}_1,\ \boldsymbol{x}_2,\ \boldsymbol{x}_3\}$ with $R_1 = 2,\ R_2 = 2,\ R_3 = 2$, assume $l = 6$ and find $\{r_1,\ r_2,\ r_3\}$.

Solution: Since $4 \le l\ (= 6) < 8$, we have the following guideline from Theorem 1:

If $R_1 R_2 \le l < R_1 R_2 R_3$, then
$r_1 = l(\text{mod})R_1, \quad r_2 = (l\backslash R_1)(\text{mod})R_2, \quad r_3 = l\backslash(R_1 R_2)$
$r_1 = 6(\text{mod})2, \quad r_2 = (6\backslash 2)(\text{mod})2, \quad r_3 = 6\backslash(2 \times 2)$
$\rightarrow \{r_1,\ r_2,\ r_3\} = \{0,\ 1,\ 1\}$.

TABLE I
DIRECT RULE GENERATION WITH THREE FUZZY VARIABLES

r_1	r_2	r_3	$l = r_1 + 2(r_2) + 4(r_3)$	μ_l
0	0	0	0	$\mu_0 = F_1^0 F_2^0 F_3^0$
1	0	0	1	$\mu_1 = F_1^1 F_2^0 F_3^0$
0	1	0	2	$\mu_2 = F_1^0 F_2^1 F_3^0$
1	1	0	3	$\mu_3 = F_1^1 F_2^1 F_3^0$
0	0	1	4	$\mu_4 = F_1^0 F_2^0 F_3^1$
1	0	1	5	$\mu_5 = F_1^1 F_2^0 F_3^1$
0	1	1	6	$\mu_6 = F_1^0 F_2^1 F_3^1$
1	1	1	7	$\mu_7 = F_1^1 F_2^1 F_3^1$

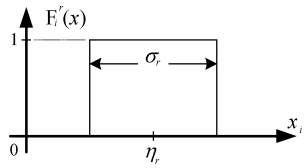

Fig. 2. BPMF.

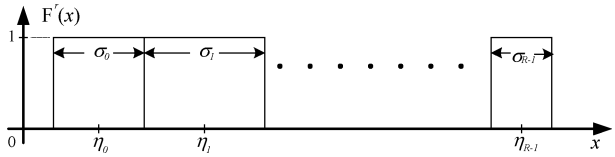

Fig. 3. *R* disjoint BPMFs for a specific x_i.

III. BLOCK PULSE MEMBERSHIP FUNCTIONS

In this section, we propose an alternative architecture of an FNN with BPMFs that are disjoint with each other and cascaded successively. Given the following fuzzy rules defined in (1)–(3), one of the BPMFs for the *r*th MF for fuzzy variable x_i $(i = 1, 2, \ldots, N)$ is defined as

$$F_i^r(x) = \begin{cases} 1, & -\dfrac{\sigma_r}{2} < x - \eta_r \le \dfrac{\sigma_r}{2} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where η_r and σ_r are the mean (or middle) and width (or variance) of the block pulse function. Fig. 2 illustrates the BPMFs.

Further we impose the following condition on successive BPMFs:

$$\eta_{r+1} - \eta_r = \frac{1}{2}\,(\sigma_{r+1} + \sigma_r)\,. \tag{6}$$

Note that σ_r's are not necessarily equal for all r. Then, all the BPMFs for fuzzy variables x_i $(i = 1, 2, \ldots, N)$ are disjoint with each other and cascaded successively. Fig. 3 shows that there are *R* disjoint and cascaded BPMFs for a specific x_i. It is noted that all the BPMFs for different fuzzy variables x_i are not necessarily disjoint. We only require that the BPMFs for a specific x_i are disjoint with each other and cascaded successively. Fig. 4 shows the BPMFs for all x_i.

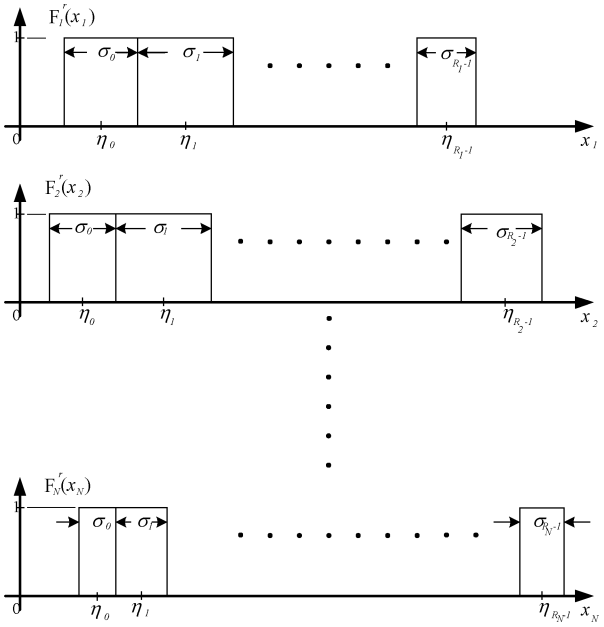

Fig. 4. Disjoint BPMFs for all x_i.

IV. FNN With BPMFs and Its Equivalence to the TL Technique

It can be shown in this section that the FNN with BPMFs is equivalent to the conventional TL technique used in hardware implementation of computer controllers [11]–[16]. The TL technique is mainly to minimize the computational effort required by the floating-point computations.

Theorem 2: Given the disjoint and cascaded BPMFs defined in (5) and (6), the fuzzy rules in layer III of FNN in Fig. 1, i.e., $\{\mu_i,\ i = 0, \ldots, L\}$, can have only one nonzero $\mu_l = 1$. Also the outputs $\{y_j, j = 1, \ldots, Z\}$ will be equal to $\{y_j = w_j^l, j = 1, \ldots, Z\}$. This is the conventional TL method for hardware implementation of computer controller.

Proof: Let $F_i^r(x)$ be the rth MF for fuzzy variable x_i. From Fig. 3, it is obvious that every fuzzy variable x_i can have only one r_ith mapping, which is a nonzero unity mapping through the disjoint and cascaded BPMFs defined in (5) and (6), i.e.,

$$F_i^{r_i}(x) = 1.$$

Assume that $\{F_1^{r_1}(x), F_2^{r_2}(x), \ldots, F_N^{r_N}(x)\}$ are the nonzero unity mappings for $\{x_1, x_2, \ldots, x_N\}$. From Theorem 1, we can find the specific fuzzy rule l that corresponds to all those nonzero unity mappings

$$l = r_1 + R_1 * (r_2) + R_1 * R_2 * (r_3) + \cdots + R_1 * R_2 * \cdots * R_{N-1} * (r_N)$$

and

$$\mu_l = \prod_{i=1}^{N} F_i^{r_i}(x_i) = 1.$$

This also implies that for all the other fuzzy rules, their μ values are all equal to zeros, i.e.,

$$\mu_k = 0, \quad \text{for } k \neq l.$$

Therefore, we have

$$\sum_{i=1}^{L} w_j^i \mu_i = w_j^l * \mu_l = w_j^l$$

and

$$\sum_{i=1}^{L} \mu_i = \mu_l = 1.$$

Thus, the output defined in (3) can be shown as

$$y_j = \sum_{i=1}^{L} w_j^i \mu_i \Big/ \sum_{i=1}^{L} \mu_i = \sum_{i=1}^{L} w_j^i \mu_i = w_j^l \mu_l = w_j^l.$$

Q.E.D.

Theorem 2 shows that the conventional TL technique is actually a special case of the modern FNN. Therefore, we can apply the recently developed training techniques in the FNN to further elaborate the conventional TL technique, such as the update of weighting factors in Theorem 2.

V. Comparison of FNN Controllers Using Traditional MFs and BPMFs

In this section, two FNN control methods utilizing traditional MFs and BPMFs are applied respectively on a water injection controller of a turbo-charged engine, and a comparison between these two FNN control methods described earlier is further presented.

Consider the following FNN for a turbo-charged engine [19] utilizing water injection control [19], [20] (Fig. 5), the water injection ratio $Water$(r,m) is the output of the water injection controller that is based on m [manifold absolute pressure (MAP)] and r [rounds per minute (RPM)]. The MAP and RPM will be fuzzified in the second layer to generate the fuzzy rules in the third layer. Each rule will be associated with a weighting factor to produce the water injection ratio $Water$(r,m). The objective is to minimize the cost function $J(r, m)$.

The cost function [1, eq. (13)] can be defined as

$$J = \frac{1}{2PZ} Tr(EE^T)$$

where P is the training data number, Z is the output number, and E is a $P * Z$ error function matrix. In this application, $Z = 1$, and the error function E can be seen as the knocking signal from the electronic control unit (ECU) of a turbo-charged engine. Thus, the cost function can be rewritten as

$$J(r, m) = \frac{1}{2P} \sum_{i=1}^{P} \text{Knock}(r, m)^2. \tag{7}$$

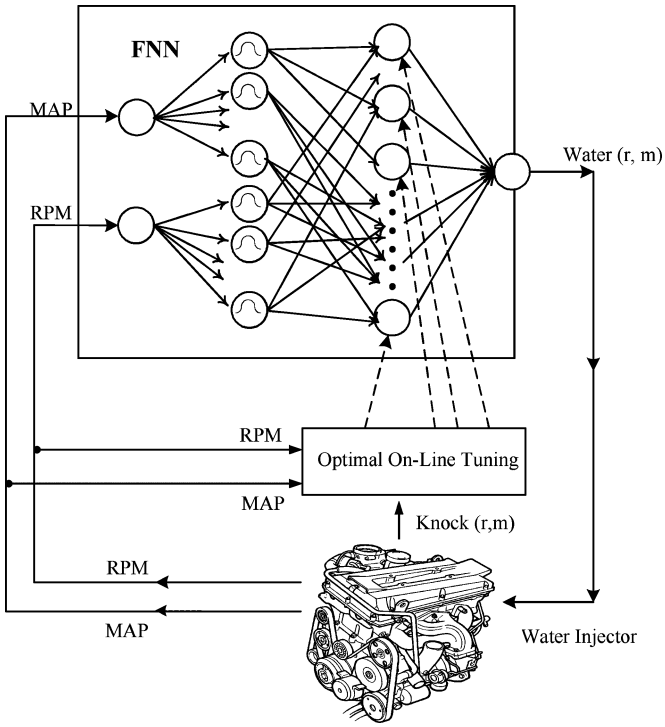

Fig. 5. FNN for a turbo-charged engine utilizing water injection control.

To update weighting matrix W, we apply the backpropagation method in (15) of [1] as follows:

$$W_{t+1} = W_t - \beta_t \frac{1}{PZ} RE$$

where β_t is the learning rate at the tth iteration, R is $L * P$ input training matrix, and L is the rule number. In this case, the backpropagation equation is

$$W_{t+1} = W_t - \beta_{\text{opt}} \frac{1}{P} R \cdot \text{Knock}(r, m) \tag{8}$$

where $\beta_{\text{opt}} = (\beta_u + \beta_l)/2$ in [1, eq. (21)].

Let MAP have five MFs and RPM have five MFs, which are shown as follows:

$$M_{\text{map}}(x_i) = \exp\left[-\left(\frac{x_i - \mu_j^m}{0.3}\right)^2\right] \tag{9}$$

$$M_{\text{rpm}}(x_k) = \exp\left[-\left(\frac{x_k - \mu_l^r}{1000}\right)^2\right] \tag{10}$$

where

1) -0.6 bar $\le x_i \le 1.6$ bar, with $\{\mu_j^m | j = 1, 5\}$ bar $= \{0.3, 0.6, 0.9, 1.2, 1.5\}$ bar,
2) 0 rpm $\le x_k \le 7000$ rpm, with $\{\mu_l^r | l = 1, 5\}$ rpm $= \{2000\ 3000\ 4000\ 5000\ 6000\}$ rpm.

Thus, we have 25 fuzzy rules. The 25 weight factors are initialized to zeros in the beginning. After applying the dynamical optimal training in [1] on the road with various driving conditions, one of trained weighting factor matrices W can be shown

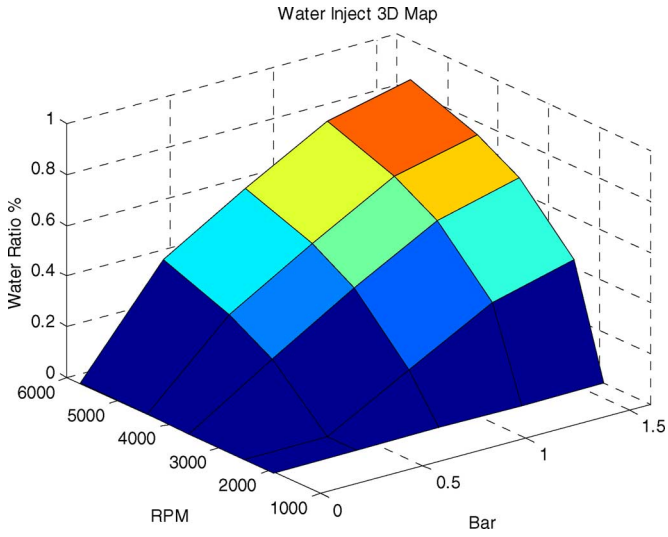

Fig. 6. Dynamic water map generated by FNN with traditional MFs.

as follows ("x" means no online training data are yet available):

$$W(j,1) = \begin{bmatrix} 0 & 0.01 & 0.10 & x & x \\ 0.02 & 0.06 & 0.14 & 0.29 & x \\ 0.05 & 0.15 & 0.26 & 0.38 & 0.56 \\ 0.09 & 0.20 & 0.44 & 0.63 & 0.65 \\ x & 0.24 & 0.48 & 0.68 & 0.71 \end{bmatrix}_{25\times 1}. \tag{11}$$

After a little longer on-road driving, a complete dynamic water map is shown in Fig. 6, which is generated by an FNN utilizing traditional overlapped MFs. Note that the training of (11) using even the dynamic optimal training algorithm is very time-consuming, which requires a more powerful CPU to complete the complete online training. This is due to the fact that the traditional MFs are overlapped with each other that will interfere with each other during the training process. It is also obvious from Fig. 6 that each surface will cover wide ranges of *rpm* and *bar*. This is a worry since this will result in a less accurate interpolated result even with the dynamical optimal training algorithm.

Secondary, the following water map (Fig. 7) is generated by using the FNN control method utilizing the proposed BPMFs. This water map is zero for every rule in the beginning and then generated by online learning. Once the learning is started, the initial means and variances are defined evenly by the number of fuzzy rules (by experience, of course), i.e., we let MAP have 15 BPMFs and RPM have 10 BPMFs, the dynamical optimal training in (8) can be applied to generate the water map. Thus, we have 150 fuzzy rules. The BPMFs of MAP and RPM can be shown as follows:

$$F_{\text{map}}(x_i) = \begin{cases} 1, & -\dfrac{\sigma_j}{2} < x_i - \eta_j \le \dfrac{\sigma_j}{2} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

$$F_{\text{rpm}}(x_k) = \begin{cases} 1, & -\dfrac{\sigma_l}{2} < x_k - \eta_l \le \dfrac{\sigma_l}{2} \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

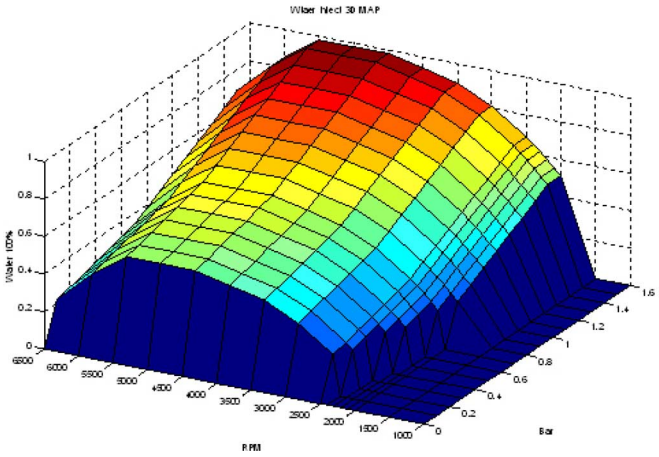

Fig. 7. Water map generated by a pure FNN with BPMFs.

where

1) $-0.6\,\text{bar} \le x_i \le 1.6\,\text{bar}$, with $\{\eta_j^m | j = 1, 15\}$ bar $= \{0.1, 0.2, 0.3, 0.4, \ldots, 1.5\}$ bar; and $\sigma_j = 0.1$ bar, for all j.
2) $0\,\text{rpm} \le x_k \le 7000\,\text{rpm}$, with $\{\eta_l^m | l = 1, 10\}$ rpm $= \{2000, 2500, 3000, \ldots, 6500\}$ rpm, and $\sigma_l = 500$ rpm, for all l.

The initialized 150 weight factors can be initialized to zeros, and after applying the dynamical optimal training (8), the final weighting factor matrix W can be briefly shown as follows ("x" means no online training data are yet available):

$$W(j,1) = \begin{bmatrix} 0 & 0 & 0 & 0 & . & . & x \\ 0 & 0 & . & . & . & . & 0.50 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ x & 0.29 & 0.34 & . & . & 0.64 & 0.74 \end{bmatrix}_{150\times 1}. \tag{14}$$

It is expected that after a short period of driving the automobile, the water map in the FNN will be similar to that in Fig. 7, provided that driving conditions vary at a wider level. It is obvious from Theorem 2 that the FNN with BPMFs adopted in Fig. 7 is equivalent to the conventional TL method [21] to generate the fixed water map. It is also obvious from Fig. 7 that the dynamic water map (with training option) is smoother than that in Fig. 6 at the cost of more rule memory, yet with less computing requirement.

According to the results after a short period of learning, Fig. 7 shows that the widths of BPMFs are adapted based on the gradient of the water map. In this paper the adaptive law for the widths of the BPMFs is that the width of BPMF is decreased more in the higher gradient area. This will, of course, increase the number of fuzzy rules so that more accuracy will be obtained. In this water inject controller case, the number of BPMFs are increased in the range of {1500–2500 rpm} after a period of online training. Hence, the initial 10 BPMFs are increased eventually to 15 BPMFs in the range of {1500–2500 rpm}.

TABLE II
COMPARISON OF FNNS WITH TRADITIONAL MFS AND BPMFS

	Traditional MF's	BPMF's
Memory space	Few(25 rules)	Large (225 rules)
Training time	Slow	Fast
Weighting Factors Dependency	Dependent on the other weighting factors	Independent of all the other weighting factors
CPU time	Slow ($O(N^{mf})$)	Fast (O(N))
Epochs for convergence	10~20	1

In Fig. 7, due to the sharp gradients within the range of {1500–2500 rpm}, the BPMFs for MAP remains nearly unchanged after training. But the BPMFs for RPM will be changed to

$$F_{\text{rpm}}(x_k) = \begin{cases} 1, & -\dfrac{\sigma_l}{2} < x_k - \eta_l \le \dfrac{\sigma_l}{2} \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

where $0\ \text{rpm} \le x_k \le 7000$ rpm, with $\{\eta_l^m | \text{l} = 1, 15\}$ rpm $= \{1300, 1500, 1700, 1900, 2100, 2300, 2500, 3000, 3500, 4000, 4500, 5000, 5500, 6000, 6500\}$ rpm, and $\sigma_l = \{200, 200, 200, 200, 200, 500, 500, 500, 500, 500, 500, 500, 500, 500, 500\}$ rpm.

Table II shows a comparison between FNN with traditional overlapped MFs (Fig. 6) and FNN with BPMFs (Fig. 7). It takes more memory space to store the fuzzy rules for FNN with BPMFs (150 MFs) than that in FNN with traditional overlapped MFs (25 MFs). But the FNN with traditional overlapped MFs will require more computing time for tuning weighting factors even with dynamical optimal training algorithm in [1]. This is due to the fact that the traditional overlapped MFs will interfere with each others' weighting factors during the training process. For online operations of FNN with traditional overlapped MFs, faster CPU with higher cost is always required. However, we know the fact from Theorem 2 that the weighting factors in FNN with BPMFs are independent with each other. Thus, the training process will converge quickly with dynamical optimal training algorithm in (8). Thus, we do not require powerful CPU in the FNN with BPMFs. It is obvious that computation time for the trainings of FNN with BPMFs and traditional MFs are $O(N)$ and $O(N^{\text{mf}})$, respectively, where mf is the number of MFs. In addition, this example shows that the FNN with nonoverlapped BPMFs takes only one epoch for convergence instead of 10–20 epochs for the FNN with traditional overlapped Gaussian MFs for convergence.

VI. CONCLUSION

This paper proposes a new BPMFs to construct a special FNN. A direct and efficient way is first developed to perform the fuzzy rule generation in FNN. Thus, we can have a one-to-one mapping between the MFs of all the input fuzzy variables and the specific number of fuzzy rules. This will not only resolve the confusing expression in all the other research articles about the fuzzy rule generation in FNN, but also the efficiency of FNN implementation is improved. Based on this fact, it is further proved that the FNN with BPMFs is actually the conventional TL technique adopted in the hardware implementation of computer controller. The impact of this new result is that

the recently developed FNN techniques can also be applied to improve the performance of the conventional TL techniques. A real application of water injection control of a turbo-charged engine using FNN controllers with traditional overlapped MFs and our newly proposed BPMFs are illustrated. Their comparison shows that the FNN with BPMFs can provide faster online update results at the cost of more memory space to store the fuzzy rules.

REFERENCES

[1] C.-H. Wang, H.-L. Liu, and C.-T. Lin, "Dynamic optimal learning rates of a certain class of fuzzy neural networks and its applications with genetic algorithm," *IEEE Trans. Syst., Man, Cybern., Cybern. B, Cybern.*, vol. 31, no. 3, pp. 467–475, Jun. 2001.

[2] C.-H. Wang, W.-Y. Wang, T.-T. Lee, and P.-S. Tseng, "Fuzzy B-spline membership function and its applications in fuzzy neural control," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 5, pp. 841–851, May 1995.

[3] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 6, pp. 1414–1427, Nov./Dec. 1992.

[4] S. Wu, M. J. Er, and Y. Gao, "A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 4, pp. 578–594, Aug. 2001.

[5] Y. G. Leu, T. T. Lee, and W. Y. Wang, "Observer-based adaptive fuzzy neural control for unknown nonlinear dynamical systems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 5, pp. 583–591, Oct. 1999.

[6] C. H. Wang, H. L. Liu, and T. C. Lin, "Direct adaptive fuzzy neural control with state observer and supervisory controller for unknown nonlinear dynamical systems," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 1, pp. 39–49, Feb. 2002.

[7] V. Boskowitz and H. Guterman, "An adaptive neuro fuzzy system for automatic image segmentation and edge detection," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 247–262, Feb. 2002.

[8] C.-S. Lee, C.-Y. Hsu, and Y.-H. Kuo, "Intelligent fuzzy image filter for impulse noise removal," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZIEEE 2002)*, vol. 1, pp. 431–436.

[9] A. V. Topalov and O. Kaynak, "Online learning in adaptive neurocontrol schemes with a sliding mode algorithm," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 31, no. 3, pp. 445–450, Jun. 2001.

[10] H. Shiraishi, S. L. Ipri, and Dong-il D. Cho, "CMAC neural network controller for fuel-injection systems," *IEEE Trans. Control Syst. Technol.*, vol. 3, no. 1, pp. 32–38, Mar. 1995.

[11] A. B. Rad, P. T. Chan, W. L. Lo, and C. K. Mok, "An online learning fuzzy controller," *IEEE Trans. Ind. Electron.*, vol. 50, no. 5, pp. 1016–1021, Oct. 2003.

[12] A. Peled and B. Liu, "A new hardware realization of digital filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-22, no. 6, pp. 456–462, Dec. 1974.

[13] N. Jain, D. Agnew, and M. Nakhla, "Two-dimensional table lookup MOSFET model," in *Proc. IEEE Integr. Conf. Comput.-Aided Design*, Santa Clara, CA, 1983, pp. 201–213.

[14] R. A. Zitar and M. H. Hassoun, "Neurocontrollers trained with rules extracted by a genetic assisted reinforcement learning system," *IEEE Trans. Neural Netw.*, vol. 6, no. 4, pp. 859–879, Jul. 1995.

[15] N. Takagi, "Powering by a table look-up and a multiplication with operand modification," *IEEE Trans. Comput.*, vol. 47, no. 11, pp. 1216–1222, Nov. 1998.

[16] S.-C. Chang, M. Marek-Sadowska, and T. T. Hwang, "Technology mapping for TLU FPGA's based on decomposition of binary decision diagrams," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 10, pp. 1226–1236, Oct. 1996.

[17] X.-Q. Liu, H.-Y. Zhang, J. Liu, and J. Yang, "Fault detection and diagnosis of permanent magnet dc motor based on parameter estimation and neural network," *IEEE Trans. Ind. Electron.*, vol. 47, no. 5, pp. 1021–1030, Oct. 2000.

[18] Saab, "Saab Workshop Information System (WIS) 900," 2001.

[19] J. A. Harrington, "Water addition to gasoline/manifold injection—Effect on combustion, emissions, performance, and knock," *SAE TRAN.*, vol. 910, Doc. 820314.

[20] J. E. Nicholls, I. A. El-Messiri, and H. K. Newhall, "Inlet manifold water injection for control of nitrogen oxides-theory and experiments," *SAE TRAN.*, vol. 780, Doc. 690018.

[21] Aquamist Water Injection, System 1 s (basis of a starter kit), System 2 c (interface with the PWM output signal), System 2 d (track of the fuel flow and inject water with a fixed w/f ratio) & System 2 s (A fully-mappable water-injection system), U.K., 2004.

Chi-Hsu Wang (M'92–SM'93–F'08) was born in Tainan, Taiwan, R.O.C., in 1954. He received the B.S. degree in control engineering from the National Chiao Tung University, Hsinchu, Taiwan, in 1976, the M.S. degree in computer science from the National Tsing Hua University, Hsinchu, in 1978, and the Ph.D. degree in electrical and computer engineering from the University of Wisconsin, Madison, in 1986.

In 1986, he was an Associate Professor in the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, where he was a Professor in 1990. He is currently a Professor in the Department of Electrical and Control Engineering, National Chiao Tung University. His current research interests include digital control, fuzzy neural network, intelligent control, adaptive control, and robotics.

Dr. Wang is currently an Associate Editor of the IEEE TRANSACTION ON SYSTEMS, MAN, and CYBERNETICS, PART B, and a Member of the Board of Governors and Webmaster of the IEEE Systems, Man, and Cybernetics Society.

Jung-Sheng Wen (M'08) was born in Taipei, Taiwan, R.O.C., in 1961. He received the B.S. and M.S. degrees from the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, in 1988 and 1991, respectively. He is currently working toward the Ph.D. degree in the Department of Electrical and Control Engineering, National Chiao-Tung University, Hsinchu, Taiwan.

Since 1992, he has been a Lecturer in the Department of Computer and Communication Engineering, Technology and Science Institute of Northern Taiwan, Taipei. His current research interests include fuzzy neural network, intelligent computer control, and embedded system design.