A Dynamic Fuzzy Neural System Design via Hybridization of EM and PSO Algorithms

Ching-Hung Lee, *Member, IANEG*, Yu-Chia Lee, and Feng-Yu Chang

***Abstract*— In this paper, we propose a modified hybridization of electromagnetism-like mechanism (EM) and particle swarm optimization (PSO) algorithms, called mEMPSO, for designing the proposed functional-link based Petri recurrent fuzzy neural system (FLPRFNS). The mEMPSO implements an instant update particle velocity strategy such that each particle updates its information instantaneously. For reducing the computational complexity, the randomly local search is replaced by PSO algorithm. In addition, the proposed FLPRFNS has the following characteristics, the consequent part is a functional-link based orthogonal basis functions and a Petri layer is adopted to eliminate the redundant fuzzy rules computation. In order to improve the ability of function approximation and have better convergence results, this study uses the functional expansion sine and cosine basis functions. Simulation on nonlinear control and nonlinear channel equalization are discussed to show the effectiveness and performance of our approach.**

***Index Terms*—Electromagnetism-like mechanism, particle swarm optimization, functional link, fuzzy neural system, Petri net.**

I. Introduction

Over the decades, the fuzzy systems and the neural networks are used successfully in many application areas [1-4]. Based on the approximation ability, many adaptive control techniques are accompanied with them for approximation of system functions or controllers. A major drawback of the existing neural fuzzy systems is that their application domain is limited to static problems due to the feed-forward network structure. Hence, a recurrent fuzzy neural network (RFNN) system is proposed to identify and control nonlinear systems [5]. Some other recurrent fuzzy neural systems also have been proposed for nonlinear systems design [6-10]. It has the ability of storing the system past information such that it is more suitable than feed-forward network for temporal problems. In literature [10], we modified the RFNN structure to propose a dynamic fuzzy neural system with functional-link based consequent part. Alternative functional-link neural networks (FLNN) have been developed in applications of function approximation, pattern recognition, and nonlinear channel equalization [11-14]. As the results of [12, 15], using the functional expansion can effectively increase the dimensionality of the input vector. Literature [12] introduces that when we select the trigonometric polynomial of orthogonal sine and cosine basis functions then the outer product terms would have better convergence results. In order to improve the ability of function approximation and have better convergence results, this study uses the functional-link neural system (FLNS) and finite impulse response (FIR) filter to construct the consequent part of the proposed FLPRFNS. Besides, a Petri net has been developed into a powerful tool for modeling, analysis, control, optimization, and implementation of various engineering systems [16-18]. In order to reduce unnecessary computation of fuzzy rules, we adopt the Petri net into the proposed FLPRFNS.

For training the neural networks and fuzzy neural systems, the evolutionary algorithms such as genetic algorithm (GA), particle swarm optimization (PSO), electromagnetism-like mechanism (EM) algorithm, and differential evolution algorithms are widely used [19-26]. The EM algorithm is a novel meta-heuristic based algorithm for optimization problem [21-24]. It simulates the electromagnetism theory by considering each particle to be an electrical charge. Subsequently, the movement of attraction and repulsion is introduced by Coulomb's law. Obviously, it has the advantages of multi-point searches and global optimization. However, the corresponding computational complexity due to the neighborhood local search should be solved [23]. On the other hand, the PSO algorithm is easy to implement and has been empirically shown to perform well on many optimization problems [25, 26]. Each member in the swarm adapts its search patterns by learning from its own experience and other members' experiences. In the PSO, a member in the swarm is called a *particle*. In this paper, we propose a novel modified hybrid learning algorithm EM and PSO (mEMPSO) where the randomly neighborhood local search of EM is replaced by PSO to improve the efficiency of EM for designing the FLPRNFS. Moreover, in order to enhance the convergent speed, an instant update particle information strategy is adopted such that each particle updates its information instantaneously.

This paper is organized as follow. Section II illustrates the structure of FLPRFNS. In Section III, the proposed mEMPSO algorithm is introduced. In Section IV, the simulation results of nonlinear control and nonlinear channel equalization are presented. Finally, Section V gives the conclusion.

This work was supported in part by the National Science Council, Taiwan, R.O.C., under contracts NSC-97-2221-E-155-033-MY3.

Ching-Hung Lee is with Department of Electrical Engineering, Yuan-Ze University, Chung-li, Taoyuan 320, Taiwan (phone: +886-3-4638800, ext: 7119; fax: +886-3-4639355, e-mail: chlee@saturn.yzu.edu.tw).

Y. C. Lee is with the Department of Electrical Engineering, Yuan-Ze University, Chung-li, Taoyuan 320, Taiwan (e-mail: s964612@mail.yzu.edu.tw).

F. Y. Chang is with the Department of Electrical Engineering, Yuan-Ze University, Chung-li, Taoyuan 320, Taiwan (e-mail: s984612@mail.yzu.edu.tw).

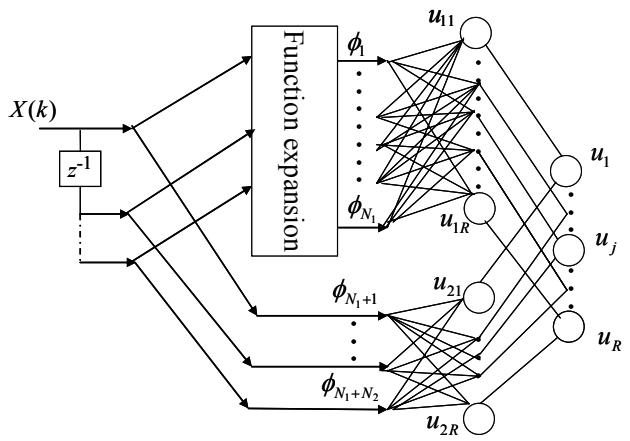

Figure 1: Diagram of combination of functional link neural system (FLNS) and FIR filter (*m*-dimensional input case).

II. Functional-Link based Petri Recurrent Fuzzy Neural System

In this section, the proposed FLPRFNS is introduced. We will first introduce the structure of the functional-link neural system and its combination of FIR filter for the consequent part of the FLPRFNS.

A. Combination of Functional-Link Neural System and FIR Filter

Literature [11] has proposed a functional link net and used the function expansion to improve the approximation performance. In order to enhance the approximate ability and performance of the fuzzy neural system, we combine the FLNS with the finite impulse response (FIR) to construct the consequent part of fuzzy rules [14]. Fig. 1 depicts the block diagram of FLNS-FIR with *m* inputs. A functional expansion block is used to expand the dimension of input to a high-dimensional space [12, 13, 15]. According to the results of [14], the FLNS with FIR filter can enhance the function mapping ability. The FLPRFNS adequately utilizes the advantages of FLNS-FIR and characteristics of FIR filter to further improve the performance.

Consider a FLNS-FIR with *m* inputs

$$X(k) = [x_1(k) \ \cdots x_m(k)]^T . \tag{1}$$

To perform the dynamic ability, the inputs are defined as

$$\mathbf{X}(k) = \begin{bmatrix} x_1(k) & \cdots & x_1(k-n+1) \\ \vdots & \ddots & \vdots \\ x_m(k) & \cdots & x_m(k-n+1) \end{bmatrix}_{m\times n} . \tag{2}$$

Obviously, a larger number *n* of time delay terms will increase the computation effort of the FLNS-FIR. In this paper, we select *n*=2 to remain the dynamic property and control the input number properly. Thus, each set of basis functions for the FLNS-FIR is shown in Fig. 1. The FLNS-FIR consists of input and trigonometric polynomial basis function. The function expansion block comprises a subset of orthogonal sine and cosine basis functions. It is defined as

$$\begin{aligned} X_1 &= [\phi_1(k)\cdots\phi_{N_1}(k)]^T \\ &= [x_1(k) \ \ \sin(\pi x_1(k)) \ \ \cos(\pi x_1(k)) \ \ \cdots \\ &\quad x_1(k-n+1) \ \ \sin(\pi x_1(k-n+1)) \ \ \cos(\pi x_1(k-n+1)) \\ &\quad \cdots \ \ x_m(k) \ \ \sin(\pi x_m(k)) \ \ \cos(\pi x_m(k)) \ \ \cdots \\ &\quad x_m(k-n+1) \ \ \sin(\pi x_m(k-n+1)) \ \ \cos(\pi x_m(k-n+1))]^T \end{aligned} \tag{3}$$

where N_1=3×*m*×*n* is the number of basis functions. The linking weights of the X_1 is

$$W_1 = \begin{bmatrix} w_{11} & \cdots & w_{1R} \\ \vdots & \ddots & \vdots \\ w_{N_1 1} & \cdots & w_{N_1 R} \end{bmatrix}_{N_1\times R} \tag{4}$$

where *R* denotes the rule number of the FLPRFNS. The FIR part is defined as

$$\begin{aligned} X_2 &= [\phi_{N_1+1}(k)\cdots\phi_{N_1+N_2}(k)]^T \\ &= [x_1(k)\cdots x_1(k-n+1)\cdots x_m(k)\cdots x_m(k-n+1)]^T \end{aligned} \tag{5}$$

where N_2=*m*×*n* is the number of basis functions. Similar to the FLNS part, the linking weights of FIR filter is

$$W_2 = \begin{bmatrix} w_{(N_1+1)1} & \cdots & w_{(N_1+1)R} \\ \vdots & \ddots & \vdots \\ w_{(N_1+N_2)1} & \cdots & w_{(N_1+N_2)R} \end{bmatrix}_{N_2\times R} . \tag{6}$$

Then, we define

$$u_{1j} = \sum_{i=1}^{N_1} w_{ij}\phi_i \tag{7}$$

$$u_{2j} = \sum_{i=1}^{N_2} w_{(N_1+i)j}\phi_{(N_1+i)} \tag{8}$$

where w_{ij} is the corresponding linking weight and ϕ_i is the basis trigonometric functional expansion of input variables. $w_{(N_1+i)j}$ and $\phi_{(N_1+i)}$ are the corresponding link weight of FIR filter and the past input variables, respectively. The output u_j of the FLNS-FIR is obtained by

$$u_j = \lambda_1 \times u_{1j} + (1-\lambda_1)\times u_{2j} \tag{9}$$

where $\lambda_1 \in [0\ 1]$.

B. Network Structure of FLPRFNS

The proposed FLPRFNS realizes a fuzzy if-then rule as follows. The *j*th fuzzy rule is represented as

Rule j: IF z_1 is A_{1j} , …, z_m is A_{mj} ,

THEN $y = u_j$

$$= \lambda_1 \times \sum_{i=1}^{N_1} w_{ij}\phi_i + (1-\lambda_1)\times \sum_{i=1}^{N_2} w_{(N_1+1)j}\phi_{(N_1+1)i}$$

where $z_j = O_{ij}^{(2)}(k-1)\cdot\theta_{ij} + O_i^{(1)}(k)$, u_j are the input and output variables, respectively; A_{ij} is the linguistic term of the precondition part with Gaussian membership function; *m* is the number of input variables; w_{ij} and $w_{(N_1+1)j}$ are the linking weights; ϕ_i denotes the basis trigonometric function. Herein, N_1 is the number of the function expansion, $\phi_{(N_1+1)i}$ denotes the basis function of past input variable, N_2 is the past input number.

Herein, we indicate the signal propagation and the operation functions of the nodes in each layer. For convenience, the multi-input-single-output case is considered here. The schematic diagram of the proposed FLPRFNS is shown in Fig. 2. In the following description, $O_i^{(l)}$ denotes the *i*th output node in layer *l*.

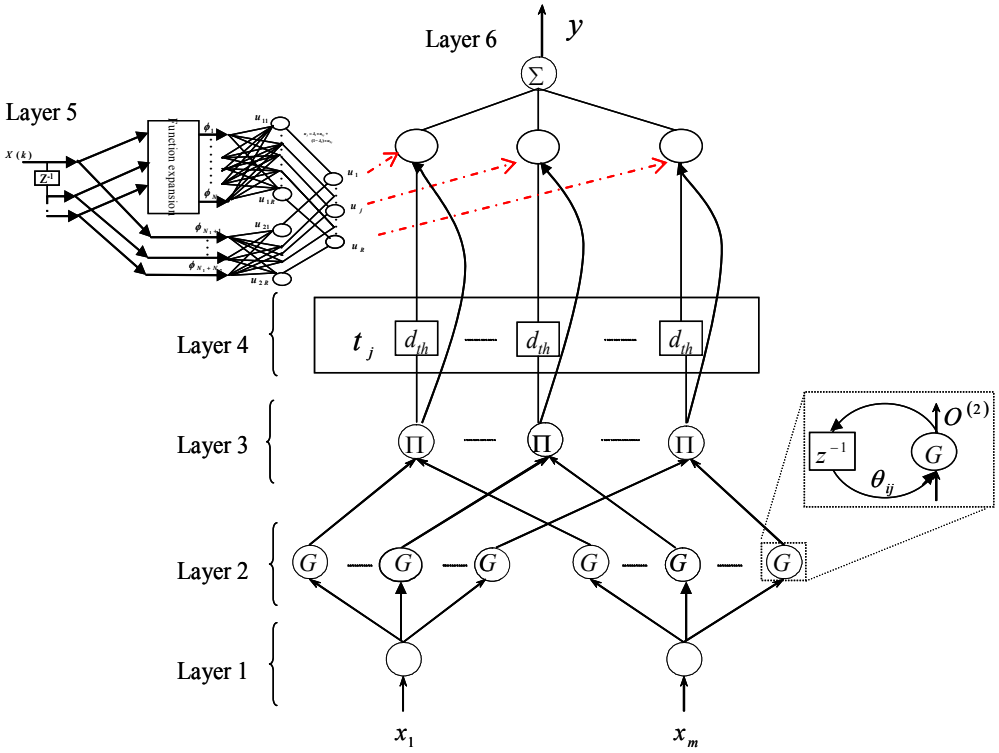

Figure 2: Structure of FLPRFNS.

Layer 1: Input Layer

For the *i*th node of layer 1, the net input and the net output are represented as

$$O_i^{(1)}(k) = x_i(k). \tag{10}$$

The x_i represents the *i*th input. The nodes in this layer only transmit the input values to the next layer directly.

Layer 2: Membership Layer

Each fuzzy set A_{ij} is constructed by the Gaussian membership function. Therefore, the output of the membership layer is defined as

$$O_{ij}^{(2)}(k) = \exp\left[-\frac{\left(z_j(k) - m_{ij}\right)^2}{\left(\sigma_{ij}\right)^2}\right], \tag{11}$$

where $z_j = O_{ij}^{(2)}(k-1)\cdot\theta_{ij} + O_i^{(1)}(k)$, m_{ij} and σ_{ij} denote the mean and variance of the *j*th term of the *i*th input of the Gaussian membership function, respectively.

Layer 3: Rule Layer

Nodes in layer 3 represent rule nodes. The product operator is adopted for calculating the firing strength of the corresponding rule. Therefore, the output function of each inference node here is

$$O_j^{(3)}(k) = \prod_i O_{ij}^{(2)}(k). \tag{12}$$

Layer 4: Petri Layer

The operation of the Petri layer is described as follows to select suitable fired nodes:

$$t_j = \begin{cases} 1, & O_j^{(3)}(k) \ge d_{th} \\ 0, & O_j^{(3)}(k) < d_{th} \end{cases}, \tag{13}$$

where t_j is the transition and d_{th} is the selected threshold value which is set between 10^{-4}~10^{-3} to eliminate redundant fuzzy rules computation as our experience.

Layer 5: TSK Layer

This layer performs the consequent part by the FLNS-FIR. The output of this layer is

$$O_j^{(5)}(k) = \begin{cases} u_j \times O_j^{(3)}(k), & t_j = 1 \\ 0, & t_j = 0 \end{cases}, \tag{14}$$

where u_j represents the *j*th output of the FLNS. Moreover, the output nodes of functional link neural network depend on the number of the fuzzy rules of the FLPRFNS.

Layer 6: Output Layer

The output layer which is acted as a defuzzifier is defined as

$$y = \frac{\sum_{j=1}^{R} O_j^{(5)}(k)}{\sum_{j=1}^{R} O_j^{(3)}(k)}, \tag{15}$$

where *R* is the number of the fuzzy rule and *y* is the output of the FLPRFNS.

As above, the tuning parameters of FLPRFNS are *m*, σ, θ, and *w*. The number of tuning parameters is $(3+4\times n)\times m\times R$, where *n*, *m*, and *R* denote the time delay, the number of inputs and the number of the fuzzy rule.

III. Modified Hybridization of Electromagnetism- like Mechanism and Particle Swarm Optimization Algorithms

This section introduces the modified hybridization of electromagnetism-like mechanism and particle swarm optimization algorithms (mEMPSO) for designing the FLPRFNS.

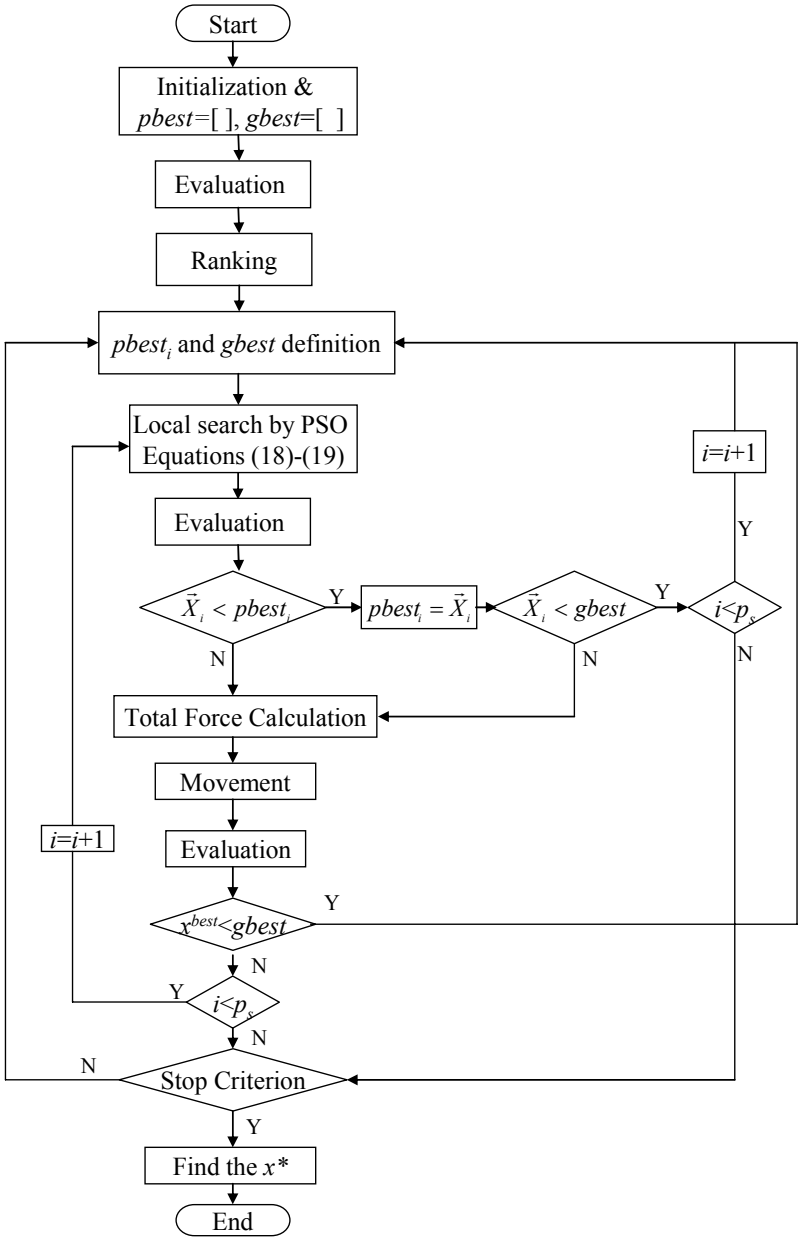

Figure 3: Flowchart of the proposed mEMPSO algorithm.

Fig. 3 depicts the flowchart of the mEMPSO algorithm. Our goal is to use the algorithm to minimize the given cost function by adjusting the parameters of the FLPRFNS. At first, the initial particles are randomly selected from the feasible region of searching space. After initialization, each particle is evaluated and ranked by the corresponding fitness function value. Here the root-mean-square-error (RMSE) is adopted. The particle having smallest RMSE value is selected to be the best particle in the group (denotes *gbest*). In addition, the best information that the particles ever reached would be memorized as population best (denotes *pbest*). Started from the second generation, each particle would update its information by using the historical best information. While each particle updates its information, the newest best particle and the newest best particle in the group would be updated. Then, the instant update particles information strategy is operated. In this strategy, all particles are updating its information instantly one by one from the current best information. After the new individual particle is defined, the *gbest* is also updated instantly. Subsequently, the EM operation phase, total force calculation and movement, is operated. Then, all particles are evaluated by the corresponding RMSE and determined the replacement of *gbest*. The particle has the smallest RMSE value is defined as x^{best}. If x^{best} is better than *gbest*, *gbest* is replaced and the velocity in next generation would be updated. The process will be stopped until the stop criterion is satisfied. Then we have the optimal particle x^*. The detailed description for the EM, PSO, and mEMPSO are introduced as below.

The mEMPSO for optimization problems is in the form of

$$\text{Minimize } f(x) \quad \text{subject to } x \in S, \tag{16}$$

where $S=\{x \in \Re^n \mid l_k \le x_k \le u_k, l_k, u_k \in \Re, k=1, \ldots, n\}$, n is problem dimension, and $f(x)$ is the objective function, u_k and l_k are the corresponding upper bound and lower bound. Each particle x represents a solution with charge.

Initialization

In this study, each particle denotes a weighting vector $[m, \sigma, \theta, w]^T$ and the mEMPSO is utilized to find the optimal

value $[m^*, \sigma^*, \theta^*, w^*]^T$. Typically, initial particles are randomly generated from a feasible solution region.

Evaluation and Ranking

To evaluate the performance of each particle in training the FLPRFNS, we select the root-mean-square-error (RMSE) as the following

$$f(x) \equiv \sqrt{\sum_{k=1}^{N} e^2(k)/N} \tag{17}$$

where *e* denotes the approximate error (or tracking error for the control problem) and *N* denotes the data number. After evaluation, all particles are ranked by the corresponding RMSE value.

***gbest* and *pbest*$_i$ Definition**

After evaluation and ranking, the particle which has the smallest RMSE value is defined as *gbest*. The best solution of *i*th particle is also defined as *pbest*$_i$.

Local Search for mEMPSO

The local search phase is used to gather the local information of each particle. In mEMPSO, the local search is implemented by PSO algorithm with an instant velocity update strategy. Herein, let $\vec{X}$ denotes the particle (i.e., $[m, \sigma, \theta, w]^T$), each particle can be updated by the following equations

$$\begin{aligned}\vec{V}_i(k+1) = \chi(\vec{V}_i(k) &+ c_1\phi_1(pbest_i(k) - \vec{X}_i(k)) \\ &+ c_2\phi_2(gbest(k) - \vec{X}_i(k)))\end{aligned} \tag{18}$$

$$\vec{X}_i(k+1) = \vec{X}_i(k) + \vec{V}_i(k+1) \tag{19}$$

where c_1, c_2 are positive constants and χ is control parameter of $\vec{V}_i$. ϕ_1, ϕ_2 are random numbers within [0 1].

Total Force Calculation

In this phase, a charge is assigned for each particle. The charge q^i of particle x^i is determined by

$$q^i = \exp\left[\frac{f(x^i) - f(x^{best})}{\sum_{k=1}^{m}[f(x^{worst}) - f(x^{best})]}\right], i = 1,2,...,m. \tag{20}$$

Then, according to the electromagnetic theory, the force is inversely propositional to the distance between two particles and directly proportional to the product of their charges. By the superposition principle, the total force vector which is exerted on x^i is computed by

$$F^i = \begin{cases} \sum_{j \neq i}^{m} (x^j - x^i) \cdot \dfrac{q^i q^j}{\left\|x^j - x^i\right\|^2} & \text{if} \quad f(x^j) < f(x^i) \\ \sum_{j \neq i}^{m} -(x^j - x^i) \cdot \dfrac{q^i q^j}{\left\|x^j - x^i\right\|^2} & \text{if} \quad f(x^j) \geq f(x^i). \end{cases} \tag{21}$$

After comparing the RMSE, the direction of the forces between the particle and the others is selected. For two particles, the one has a smaller RMSE value attracts the other one. On the other hand, the particle with larger fitness value repels the others. Detailed descriptions can be found in literature [21, 23].

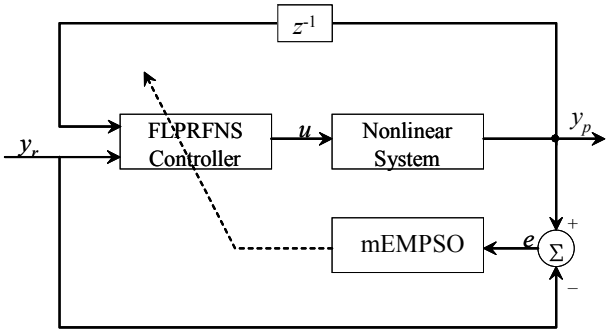

Figure 4: The dynamic system control configuration with FLPRFNS controller.

Movement

After determining the total force vector F^i, particle x^i moves in the direction of the total force by a random step length, that is

$$x^i = x^i + \lambda \frac{F^i}{\left\|F^i\right\|}(RNG) \quad i = 1, 2, ..., m \tag{22}$$

where λ=random(0,1) and

$$RNG = \begin{cases} u_k - x_k^i & \text{if} \quad F_k^i > 0 \\ x_k^i - l_k & \text{if} \quad F_k^i \leq 0 \end{cases} \quad k = 1, 2, ..., n.$$

Stop Criterion

In general, the stop criterion can be chosen as maximum generations or specific performance. In this study, the maximum generations is used to be the stop criterion.

IV. Simulation Result

To demonstrate the performance of the FLPRFNS and the mEMPSO, the simulations regarding nonlinear control and signal processing are introduced. The FLPRFNS with mEMPSO is applied to nonlinear control system and nonlinear time-varying channel equalization.

A. Nonlinear System Control

We consider the tracking control of nonlinear system with

$$y_p(k+1) = \frac{y_p(k)}{1 + y_p^2(k)} + u(k). \tag{23}$$

The reference model is described as

$$y_r(k+1) = 0.6y_r(k) + r(k) \tag{24}$$

where

$$r(k) = \begin{cases} \sin\dfrac{2\pi k}{10} + \sin\dfrac{2\pi k}{25}, & k \leq 100 \\ \sin\dfrac{2\pi k}{25}, & 100 < k \leq 300 \end{cases}.$$

Note that system state is y_p and tracking trajectory vector is y_r. Fig. 4 shows the dynamic system control configuration by FLPRFNS controller and mEMPSO algorithm. The inputs of FLPRFNS controller are y_p and y_r and the output is control signal u. The corresponding RMSE function of tracking error is defined as

$$\text{RMSE} = \left(\sum_{k=1}^{300}(y_r(k+1) - y_p(k+1))^2/300\right)^{1/2}. \tag{25}$$

To show the efficiency and effectiveness of the mEMPSO, we have the comparison results of EM, PSO, EMPSO, and GA algorithms. For the mEMPSO algorithm, the following parameters are chosen

- Maximum generations: 300
- Particle number: 28
- Control constant: 1
- Positive constant C_1: 2
- Positive constant C_2: 2

The initial parameters of the FLPRFNS are chosen randomly between [-1, 1] and the network structure is

- Network structure: 2-10-5-5-5-1
- Parameter number of FLPRFNS: 110
- Rule number of FLPRFNS: 5

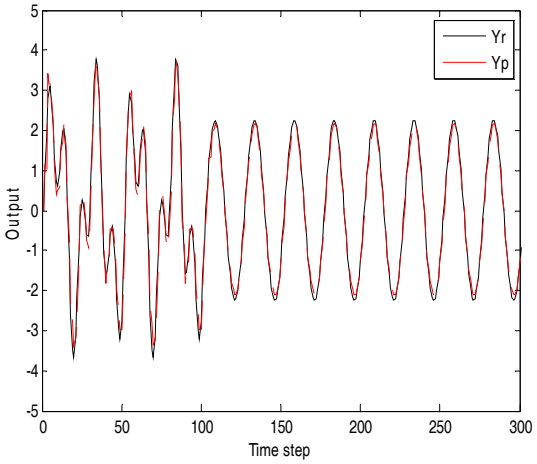

Figure 5: The system trajectories after 300 generations (solid line: desired trajectory; dashed line: system actual output).

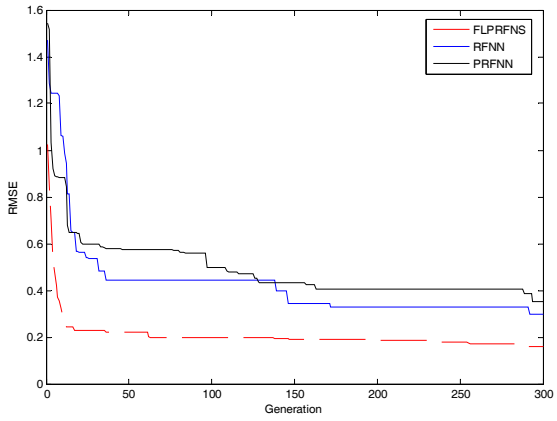

Figure 6: The comparison results of different network structure with the same number of turning parameters (the number of turning parameters: 154).

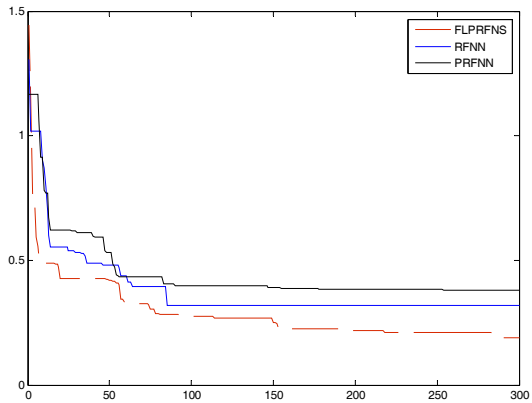

Figure 7: The comparison results of different networks with the same rule number (R=5).

Fig. 5 shows the system trajectories after 300 generations (solid line: desired trajectory; dashed line: system actual output). Obviously, the system output y_p can follow the desired output y_r after training in better performance. That is, the FLPRFNS with mEMPSO performs well for the nonlinear control. Fig. 6 shows the comparison results of different networks with the same number of turning parameters and Fig. 7 shows the comparison results of different network structure with the same fuzzy rule number. We can observe that whether in the same number of turning parameters or in the same rule number, the FLPRFNS has better training results (in RMSE) than RFNN and PFRNN. In addition, the convergent speed of the FLPRFNS is also faster than the RFNN and PFRNN. That is, the FLPRFNS can obtain better performance using less fuzzy rules for nonlinear control problem. Detailed comparison results are also shown in Table I.

For the performance comparison of algorithm, Fig. 8 shows the comparison results of tracking error RMSE (dashed line: mEMPSO, solid blue line: EMPSO, solid black line: EM, solid pink line: PSO and solid green line: GA). Fig. 9 shows the comparison results in RMSE versus evaluations. As above results and discussion of Figs. 8 and 9, we can conclude that the hybrid learning algorithm mEMPSO has good performance in convergent speed and accuracy.

Table I: The comparison results of different network structure.

Structure.	Rule number	The number of turning parameters	RMSE
PRFNN	5	35	0.382
	9	63	0.351
	10	70	0.337
	15	105	0.286
	16	112	0.273
	22	154	0.398
RFNN	5	35	0.321
	9	63	0.296
	10	70	0.271
	15	105	0.239
	16	112	0.225
	22	154	0.365
FLPRFNS	3	66	0.233
	5	110	0.192
	7	154	0.152

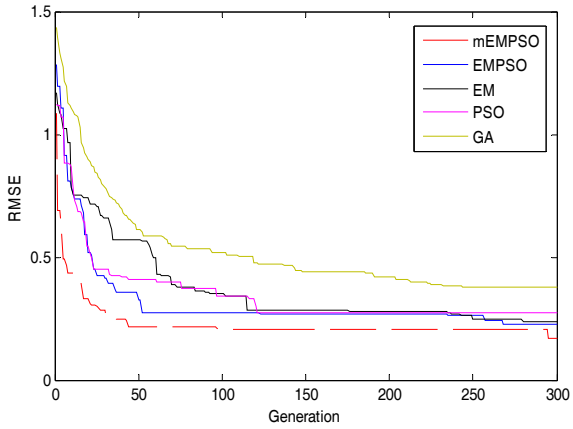

Figure 8: Comparison results of tracking error RMSE (dashed line: mEMPSO, solid blue line: EMPSO, solid black line: EM, solid pink line: PSO and solid green line: GA).

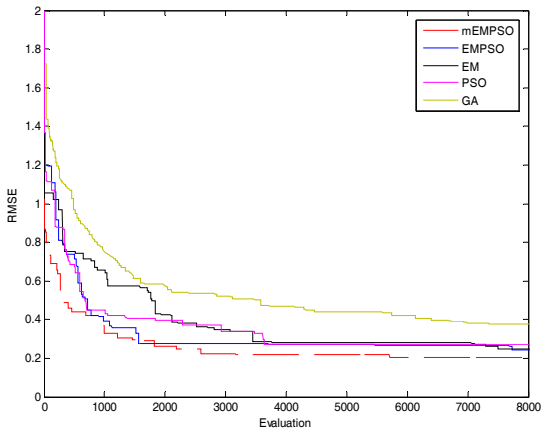

Figure 9: Comparison results in RMSE versus evaluations for nonlinear control.

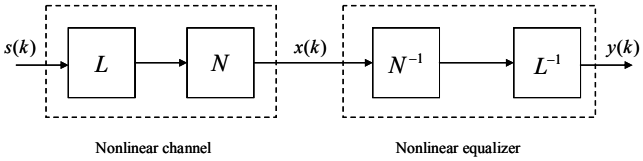

Figure 10: Diagram of nonlinear equalization model for nonlinear channel.

B. Nonlinear Channel Equalization

For a general nonlinear channel in digital communication system as shown in Fig. 10, where L and N denote the linear and nonlinear channels, respectively. In Fig. 10, the nonlinear channel consists of linear channel cascaded nonlinear part in nature. On the other hand, linear and nonlinear distortions are concurrent. Therefore, we can obtain that nonlinear equalizer should be made up of linear and nonlinear filters to compensate for the linear and nonlinear distortions. It is depicted in nonlinear equalizer part in Fig. 10, where N^{-1} denotes the nonlinear inverse filter and L^{-1} denotes the linear inverse filter. Therefore, the channel characteristic represents random temporal fluctuations by the time-varying amplitude factor. The received signal can be described as follows

$$\hat{x}(k) = c_1 s(k) + c_2 s(k-1) + \ldots + c_p s(k-p+1) + n(k) \quad (26)$$

where $s(k)$ is transmitted signal, and $\hat{x}(k)$ denotes the channel state; c_l, l=1, 2, …, p, are time-varying amplitude factor, and p is the channel order.

The nonlinear channel equalization is a technique used to combat some imperfect phenomenon in high-speed data transmission over channels [27]. Fig. 11 shows the block diagram of a communication system that is subject to inter-symbol interference (ISI) and additive white Gaussian noise (AWGN). The transmitted input symbols $s(k)$ is an independent and identically distributed discrete-time random processes taking its value {-1, +1}. The signal is sent through the channel.

In real communication systems, the channel is too dispersive to cause interference between successive signal samples (inter-symbol interference). It will complicate reliable transmission and reception. Let $\hat{x}(k)$ denotes the channel output. The channel function can be described as [28]

$$\hat{x}(k) = f\big(s(k), s(k-1), \cdots, s(k-p+1)\big) \quad (27)$$

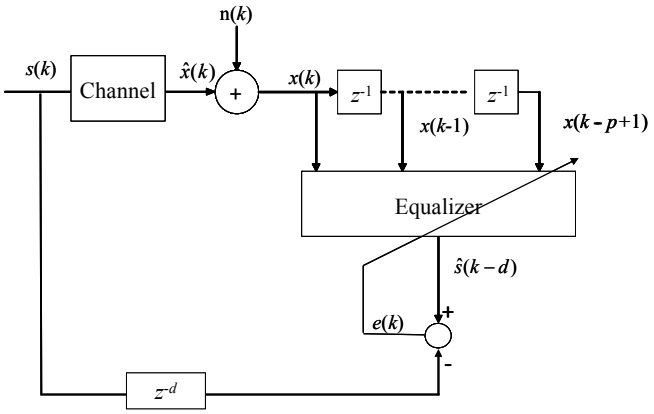

Figure 11: Block diagram of adaptive equalizer.

where p is the channel order. Generally, f is a nonlinear function of past transmitted signals, and the channel changes slowly but significantly over time; therefore, a nonlinear channel equalizer with adaptation ability is needed. The channel state is defined as $\hat{\mathbf{x}}(k) \equiv [\hat{x}(k), \cdots, \hat{x}(k-n+1)]^T$. At receiving terminal, the inter-symbol interference and nonlinear distortion are introduced by the channel; received signals $x(k)$ are also assumed to be corrupted by a additive noise $n(k)$, that is

$$x(k) = \hat{x}(k) + n(k) \quad (28)$$

where $n(k)$ is an AWGN which is assumed to be zero mean.

The function of the equalizer is to re-construct the transmitted signal, $s(k-d)$ (d denotes the decision delay), from the observed information sequence, $x(k)$, …, $x(k-p+1)$, from which greater speed and reality can be achieved. Thus, the mathematical representation of equalizer is

$$\hat{s}(k-d) = \psi(x(k), x(k-1), \cdots, x(k-p+1)) \quad (29)$$

where $\psi : \Re^p \to \{1, -1\}$. Thus, a correct decision by the equalizer is

$$\hat{s}(k-d) = s(k-d). \quad (30)$$

Based on the category of $s(k-d)$ (i.e., ± 1), the channel states $\hat{x}(k)$ can be partitioned into two classes [29]

$$X^+ = \{\hat{x}(k) \mid s(k-d) = 1\}, \quad (31)$$

$$X^- = \{\hat{x}(k) \mid s(k-d) = -1\}. \quad (32)$$

The numbers of elements in X^+ and X^- are denoted as n_s^+ and n_s^-, respectively. The probabilities for $s(k-d)$=1 and $s(k-d)$=-1 are the same, which means $n_s^+ = n_s^- = n_s / 2$ where n_s is the total number of channel state. Besides, the channel states in X^+ and X^- are denoted as $\hat{x}_i^+ (i = 1, \cdots, n_s^+)$ and $\hat{x}_i^- (i = 1, \cdots, n_s^-)$, respectively.

Suppose that the channel order is p=2 in the nonlinear channel function. For a time-varying channel, the coefficients of the channel c_i, i=0, …, n, are unknown. The nonlinear time-varying channel model is described as [28, 30]

$$x(k) = c_1 s(k) + c_2 s(k-1) - H(k) + n(k) \quad (33)$$

where c_1 and c_2 are time-varying coefficients, and $H(k)$ is co-channel Interference (CCI) which is described as

$$H(k) = \lambda[c_{11}(k) + c_{12}(k-1)] \quad (34)$$

where $c_{11}(k)$ and $c_{12}(k)$ are co-channel time-varying coefficients.

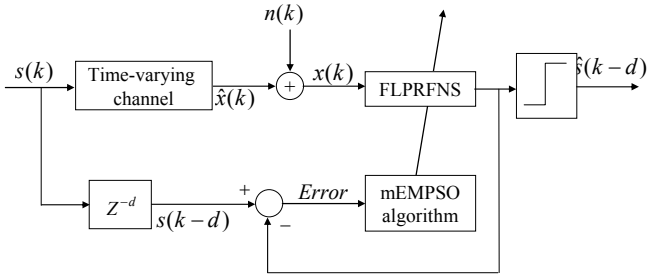

Figure 12: Block diagram of adaptive equalizer using FLPRFNS.

The time-varying coefficients c_1 and c_2 are simulated by using the second-order Markov model. It is also called second-order Butterworth low-pass filter which is derived by white Gaussian noise source [31]. In the following simulations, Matlab is used to generate a second-order low-pass digital Butterworth filter with cutoff frequency 0.1, time-varying channel coefficients are provided. Coefficients $c_1(k)$ and $c_2(k)$ are selected around 1 and 0.5, respectively. The input to Butterworth filter is a white Gaussian sequence code for time-varying coefficients with length of 1000.

Herein, we use the FLPRFNS to be the adaptive equalizer for time-varying channel equalization. As shown in Fig. 12, we use the FLPRFNS filter to construct the equalizer and use the error to update the parameters of FLPRFNS filter which can achieve the adaptive equalizer. In our simulations, we choose the independent input sequence $s(k)$ which consists of 2000 symbols. The first 1000 symbols are used for training and the remaining 1000 are used for testing. After training, the parameters of the PRFNN, RFNN and FLPRFNS filters are fixed, and then the testing is performed. We compare two examples (with CCI and without CCI) among these three types of PRFNN, RFNN, and FLPRFNS filters.

At first, we consider the CCI free case (i.e. $H(k)$=0). Hence, we assume that the time-varying channel is generated by (33) and we choose 5 rules to construct the FLPRFNS filter. Next, we consider the time-varying channel with the co-channel for CCI [28], that is

$$H(k) = 0.9 \cdot (c_{11}(k) + c_{12}(k-1))^3 \qquad (35)$$

where the nominal values are $c_{11}(k)$=1 and $c_{12}(k)$=0.5.

In order to show the effectiveness of FLPRFNS, the comparison results between FLPRFNS, RFNN, and PRFNN are introduced. In addition, two cases are considered, channel without CCI (CCI free) and channel with CCI.

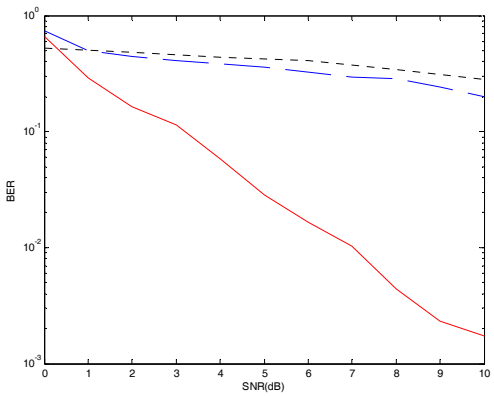

Figure 13: Comparison results of nonlinear time-varying channel where the rule number of RFNN and PRFNN is the same as FLPRFNS without CCI (solid-line: FLPRFNS, dashed-line: RFNN, and dotted-line: PRFNN).

For CCI free cases, Fig. 13 shows the comparison results of nonlinear time-varying channel where the rule numbers of FLPRFNS, RFNN, and PRFNN are the same (R=5), solid-line: FLPRFNS, dashed-line: RFNN, and dotted-line: PRFNN. In addition, Fig. 14 shows the comparison results of nonlinear time-varying channel where the turning parameter number of RFNN and PRFNN is similar to FLPRFNS without CCI (FLPRFNS: 110, RFNN: 112, PRFNN: 112), solid-line: FLPRFNS, dashed-line: RFNN and dotted-line: PRFNN. The comparisons of network structure and bit error rate (BER) are shown in Table II. Obviously, the performance using our approach is also better than RFNN and PRFNN (FLPRFNS has the smaller BER value).

For the channel with CCI cases, Fig. 15 shows the comparison results of FLPRFNS, RFNN, and PRFNN with the same fuzzy rule (rule number is 5), solid-line: FLPRFNS, dashed-line: RFNN, and dotted-line: PRFNN. Fig. 16 shows the comparison results of FLPRFNS, RFNN, and PRFNN with the similar tuning parameters (FLPRFNS: 110, RFNN: 112, PRFNN: 112), solid-line: FLPRFNS, dashed-line: RFNN and dotted-line: PRFNN. As above discussion, we can see that our approach results better performance and has advantages of fewer adjustable parameters and smaller BER value even the channel having CCI noise.

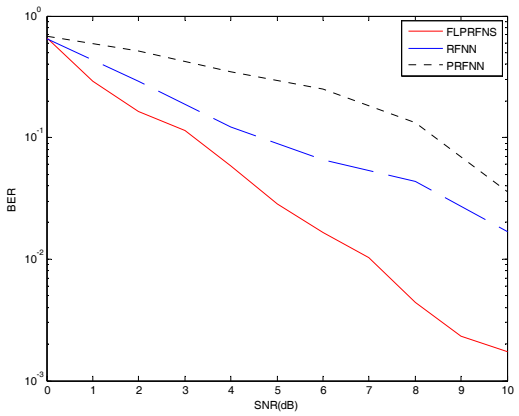

Figure 14: Comparison results of nonlinear time-varying channel where the turning parameter number of RFNN and PRFNN is similar to FLPRFNS without CCI (solid-line: FLPRFNS, dashed-line: RFNN and dotted-line: PRFNN).

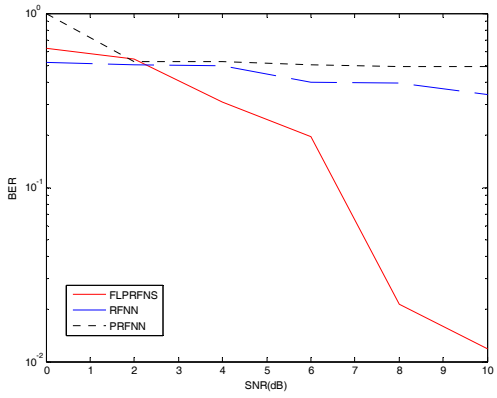

Figure 15: Comparison results of nonlinear time-varying channel where the rule number of RFNN and PRFNN is the same as FLPRFNS with CCI (solid-line: FLPRFNS, dashed-line: RFNN and dotted-line: PRFNN).

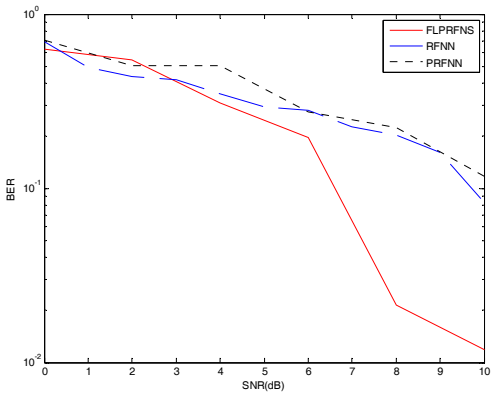

Figure 16: Comparison results of nonlinear time-varying channel where the turning parameter number of RFNN and PRFNN is similar to FLPRFNS with CCI (solid-line: FLPRFNS, dashed-line: RFNN and dotted-line: PRFNN).

Table II: Comparison results of network structure, rule number, parameter number, and BER for nonlinear time-varying channel (SNR=10 dB).

	PRFNN		RFNN		FLPRFNS
Network structure	2-10-5-1	2-32-16-1	2-10-5-1	2-32-16-1	2-10-5-5-5-1
Rule number	5	16	5	16	5
Parameter number	35	112	35	112	110
BER-CCI free	0.2882	0.0356	0.1989	0.0168	0.0017
BER- with CCI	0.5103	0.1171	0.3427	0.0823	0.0119

V. Conclusion

In this paper, a hybrid learning algorithm mEMPSO with an instant update strategy is proposed to design the functional-link based Petri recurrent fuzzy neural system (FLPRFNS). In addition, the FLPRFNS uses a functional expansion and FIR filter for the consequent part to enhance the performance and effectiveness. For training the FLPRFNS, the mEMPSO combines the advantages of EM and PSO algorithms with instant update stragtegy. Thus, it has the properties of multipoint search, global optimization, and faster convergence. It does not need any system gradient information and each particle obtains the newest information instantaneously for optimization. Simulations of nonlinear system control and nonlinear time-varying channel equalization are shown to demonstrate the effectiveness, accuracy, and better convergent performance of the FLPRFNS and mEMPSO algorithm.

References

[1] C. F. Juang and C. T. Lin, "An On-line Self-constructing Neural Fuzzy Inference Network and Its Applications," *IEEE Trans. on Fuzzy Systems*, Vol. 6, No. 1, pp. 12-32, 1998.

[2] D. H. Kim, "Parameter Tuning of Fuzzy Neural Networks by Immune Algorithm," *IEEE Int. Conf. on Fuzzy Systems*, Vol. 1, pp. 408-413, May, 2002.

[3] C. H. Lee, "Stabilization of Nonlinear Nonminimum Phase Systems: An Adaptive Parallel Approach Using Recurrent Fuzzy Neural Network," *IEEE Trans. on Systems, Man, Cybernetics- Part: B*, Vol. 34, No. 2, pp. 1075-1088, 2004.

[4] C. H. Lee and M. H. Chiu, "Adaptive Nonlinear Control Using TSK-type Recurrent Fuzzy Neural Network System," *Lecture Notes in Computer Science*, Vol. 4491, pp. 38-44, 2007.

[5] C. H. Lee and C. C. Teng, "Identification and Control of Dynamic Systems Using Recurrent Fuzzy Neural Networks," *IEEE Trans. on Fuzzy Systems*, Vol. 8, No. 4, pp. 349-366, 2000.

[6] C. J. Lin, C. Y. Lee, and C. C. Chin, "Dynamic Recurrent Wavelet Network Controllers for Nonlinear System Control," *Journal of The Chinese Institute of Engineers*, Vol. 29, No. 4, pp. 747-751, 2006.

[7] C. J. Lin and Y. J. Xu, "A Novel Evolution Learning for Recurrent Wavelet-based Neuro-fuzzy Networks," *Soft Computing Journal*, Vol. 10, No. 3, pp. 193-205, 2006.

[8] C. F. Juang, "A TSK-type Recurrent Fuzzy Network for Dynamic Systems Processing by Neural Network and Genetic Algorithms," *IEEE Trans. on Fuzzy Systems*, Vol. 10, No. 2, pp. 155-170, 2002.

[9] P. A. Mastorocostas and J. B. Theocharis, "A Recurrent Fuzzy-neural Model for Dynamic System Identification," *IEEE Trans. on Systems, Man, Cybernetics- Part: B*, Vol. 32, No. 2, pp. 176-190, 2002.

[10] C. H. Lee, J. H. Liang, and Y. C. Lee, "Nonlinear Systems Design by a Novel Fuzzy Neural System via Hybridization of EM and PSO Algorithms," *Lecture Notes in Engineering and Computer Science: Proceedings of The International MultiConference of Engineers and Computer Scientists 2010*, IMECS 2010, 17-19 March, 2010, Hong Kong, pp. 19-24.

[11] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*. Reading, MA: Addison-Wesley, 1989.

[12] J.C. Patra, R.N. Pal, "A Functional Link Artificial Neural Network for Adaptive Channel Equalization," *Expert System with Signal Processing*, Vol.43, Issue 2, pp.181-195, May. 1995.

[13] H. Zhao and J. Zhang, "Functional Link Neural Network Cascaded with Chebyshev Orthogonal Polynomial for Nonlinear Channel Equalization," *Expert System with Signal Processing*, Vol. 88, Issue 8, pp. 1946-1957, Aug 2008.

[14] H. Zhao and J. Zhang, "Adaptively Combined FIR and Functional Link Artificial Neural Network Equalizer for Nonlinear Communication Channel," *IEEE Trans. on Neural Network*, Vol. 20, No. 4, pp. 665-674, Apr. 2009.

[15] C. H. Chen, C.J. Lin, and C. T. Lin, "A Functional-Link-Based Neurofuzzy Network for Nonlinear System Control," *IEEE Trans. on Fuzzy Systems*, Vol. 16, No. 5, pp. 1362-1377, Oct. 2008.

[16] R. David and H. Alla, "Petri Nets for Modeling of Dynamic Systems—A survey," *Automatica*, Vol. 30, No. 2, pp. 175–202, Feb. 1994.

[17] K. Hirasawa, M. Ohbayashi, S. Sakai, and J. Hu, "Learning Petri Network and Its Application to Nonlinear System Control," *IEEE Trans. Syst., Man, Cybern.*, Vol. 28, No. 6, pp. 781–789, Dec. 1998.

[18] S. I. Ahson, "Petri Net Models of Fuzzy Neural Networks," *IEEE Trans. Syst. Man Cybern.*, Vol. 25, No. 6, pp. 926–932, Jun. 1995.

[19] C. H. Lee and M. H. Chiu, "Recurrent Neuro-fuzzy Control Design for Tracking of Mobile Robots via Hybrid Algorithm," *Expert Systems with Applications*, Vol. 36, No. 5, pp. 8993-8999, July 2009.

[20] J. Ilonen, J. K. Kamarainen, and J. Lampinen, "Differential Evolution Training Algorithm for Feed-Forward Neural Networks," *Neural Processing Letters*, Vol. 17, No. 1, pp. 93-105, 2003.

[21] I. Birbil and S. C. Fang, "An Electromagnetism-like Mechanism for Global Optimization," *Journal of Global Optimization*, Vol. 25, No.3, pp. 263-282, 2003.

[22] S. I. Birbil, S. C. Fang, and R. L. Sheu, "On the Convergence of A Population-based Global Optimization Algorithm," *Journal of Global Optimization*, Vol. 30, No.2, pp. 301-318, 2004.

[23] C. H. Lee and F. K. Chang, "Recurrent Fuzzy Neural Controller Design for Nonlinear Systems Using Electromagnetism-like Algorithm," *Far East Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 1, No. 1, pp. 5-22, 2008.

[24] C. H. Lee and Y. C. Lee, "An Improved Electromagnetism-like Algorithm for Neural Fuzzy Systems Design" *The 16th National Conference on Fuzzy Theory and Its Applications*, Taoyuan, Taiwan, December, 2008.

[25] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micromachine Human Sci.*, Nagoya, Japan, 1995, pp. 39–43.

[26] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks*, 1995, pp. 1942–1948.

[27] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1985.

[28] Y. C. Lin, *Type-2 Fuzzy Neural Network System and Its Applications in Signal Processing, Nonlinear System Identification, and Control*, Master Thesis of Yuan-Ze University, Taiwan, 2004.

[29] S. Chen, B. Mulgrew, and S. McLaughlin, "A Clustering Technique for Digital Communication Channel Equalization Using Radial Basis Function Network," *IEEE Trans. on Neural Networks*, Vol. 4, No. 2, pp.

570-579, 1993.
[30] A. Q. Liang and J. M. Mendel, "Overcoming Time-varying Co-channel Interference Using Type-2 Fuzzy Adaptive Filters," *IEEE Trans. on Circuits and Systems*, Vol. 47, No. 12, pp. 1419-1428, 2000.
[31] A. Q. Liang and J. M. Mendel, "Equalization of Nonlinear Time-varying Channels Using Type-2 Fuzzy Adaptive Filters," *IEEE Trans. on Fuzzy Systems*, Vol. 8, No. 5, pp. 551-563, 2000.

Ching-Hung Lee was born in Taipei County, Taiwan, R.O.C., in 1969. He received the B.S. and M.S. degree in Control Engineering from the National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1992 and 1994, respectively, and the Ph.D. degree in Electrical and Control Engineering from the same University, in 2000. He is currently an Associate Professor of the Department of Electrical Engineering at Yuan Ze University. He is the Director for the Intelligent Control and Applications Laboratory. He is a chapter author of Recurrent Neural Network (I-Tech, 2008). He has authored numerous published journal papers in the area of intelligent control systems and applications. Dr. Lee received the Wu Ta-Yu Medal and Young Researcher Award in 2008 from the National Science Council, R.O.C. He also received the 2009 Youth Automatic Control Engineering Award from Chinese Automatic Control Society. He is also the recipient of the 2009 YZU Award Excellence in Research and 2009 YZU Award Excellence in Teaching from Yuan Ze University, R.O.C. His main research interests are fuzzy neural systems, neural network, evolutionary algorithm, signal processing, nonlinear control systems, image processing, and robotics control.

Yu-Chia Lee was born in Taipei City, Taiwan, R.O.C., in 1982. He received the B.S. degree in aerospace engineering at Tamkang University, Taipei, Taiwan, R.O.C., in 2003. He received the M.S. degree in electrical engineering at Yuan Ze University, Chung Li, Taiwan, R.O.C., in 2010. His research interests include evolutionary algorithm and intelligent control.

Feng-Yu Chang was born in Hsinchu, Taiwan, R.O.C., in 1987. He received the B.S. degree in electrical engineering at Yuan Ze University, Chung Li, Taiwan, R.O.C., in 2009. He is currently working toward the Master degree in electrical engineering at the same university. His research interests include pattern recognition, intelligent system design, and evolutionary algorithm.