J. Basic. Appl. Sci. Res., 3(5)302-309, 2013

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

Nonlinear Process Identification Using Fuzzy Wavelet Neural Network Based on Particle Swarm Optimization Algorithm

Alireza Esmaili[1], Mehdi Shahbazian[2,*], Bahman Moslemi[2]

[1]Department of Instrumentation and Automation, Petroleum University of Technology, Ahwaz, Iran
[2]Assistant Professor, Ahwaz Faculty of Petroleum, PUT, Ahwaz, Iran

ABSTRACT

In this paper, the particle swarm optimization (PSO) is proposed to train fuzzy wavelet neural network (FWNN) for process system identification. The structure of FWNN is based on the fuzzy rules including wavelet functions in the consequent parts of rules. In order to improve the Identification accuracy and general capability of the FWNN system, an efficient particle swarm optimization (PSO) is used to adjust the parameters of dilation, translation, weights, and membership functions parameters. Two dynamic system identification case studies are presented to demonstrate the design flexibility and usefulness of this proposed approach. Simulation result show that the performance of PSO is superior to that the existing methods such as GA or BFGS.
KEYWORDS: Identification, Fuzzy Wavelet Neural Network, Particle Swarm Optimization, nonlinear process

1. INTRODUCTION

System identification implicates finding a relation between the input and output of the system [1], [2]. In different engineering, models are required for the design of new processes and for the analysis of existing processes[3]. As a result of rapid industrial developments, soft computing methodologies, such as fuzzy logic, neural network, and evolutionary technique, such as GA and PSO are getting more and more important and received a lot of attention in recent years.

In recent years, wavelets have become very popular and have been applied in many scientific and engineering research areas such as system identification, signal processing and function approximation [10]. They have very important properties such as time-frequency localization property. With this property, wavelets can capture global (low frequency) and local (high frequency) behavior of any function easily [11]. Wavelet neural networks (WNN) which combine neural networks with wavelet functions are also used in function approximation and system identification problems [12, 13]. In addition to having the good properties of NNs, WNNs can converge quickly and give high precision with reduced network size because of the time–frequency localization properties of wavelets [1, 8, 9].

The role of artificial neural networks in the present world applications is gradually increasing and faster algorithms are being developed for training NNs, WNNs and neuro-fuzzy system. In general, back propagation is a widely used method for training neural networks [14, 15].Gradient descent, conjugate gradient descent, resilient, BFGS quasi-Newton, one-step secant, Levenberg-Marquardt and Bayesian regularization are all different forms of the back propagation training algorithm [16, 17]. For all these algorithms storage and computational requirements are different, some of these are good for pattern recognition and others for function approximation It is difficult to find a particular training algorithm that is the best for all applications under all conditions all the time [18].

To improve the identification performance of nonlinear systems based on ANN, evolutionary algorithms (EA) (such as GA, PSO) have been used in literature. Unlike other optimization techniques, the EAs are a population-based search algorithms, which work with a population of chromosomes or particles that represent different potential solutions. Therefore, EAs have inherent parallelism that improves their exploration and the optima can be located more precisely. Some researchers have applied PSO technique [23, 24] to identify the nonlinear systems with higher convergence rate. PSO is a population based algorithm which ensures the convergence of model parameters to the global optimum. PSO offers faster convergence during training and computationally involves low complexity as compared to GA.

This paper investigates the performance of PSO algorithms in the training of FWNN structure, and for identification of nonlinear system.

The organization of the present work is embodied in the following section. the model structure of the FWNN are explained in Section II. The training agorithm and parameter update rules for the FWNN are introduced in

***Corresponding Author:** Mehdi Shahbazian, Assistant Professor, Ahwaz Faculty of Petroleum, PUT, Ahwaz, Iran.
Email (shahbazian@put.ac.ir) , mobile(09161110625) , telephone and Fax (+986115552709).

Section III. To illustrate and compare the performance of the PSO, two identification simulation examples are provided in section IV. Finally, a brief conclusion is drawn in Section V.

I. Fuzzy wavelet neural network model structure

The FWNN models combine Sugeno fuzzy system with wavelet functions. In a Sugeno fuzzy model, input space is divided into fuzzy regions and each region shows a fuzzy membership functions for an input variable [26]. The consequents of fuzzy rules are represented by either a constant or a linear function of inputs. In FWNN structure, constant or linear functions in consequent part of the rules are substituted with wavelet functions in order to increase computational power of neuro-fuzzy systems due to multi resolution property of wavelet. This property is very useful for function identification problems. The wavelets can capture global (low frequency) and local (high frequency) behavior of any function easily. This characteristic causes the proposed FWNN to be of the advantages of fast convergence, easy training and high accuracy. The rules are in following form:

$$IF\ x_1\ is\ A_{11}\ AND\ x_2\ is\ A_{21}\ THEN\ \Psi_{11} \tag{1}$$

Where X_1 and X_2 are input variables, A_{11} and A_{21} are Gaussian type membership functions and Ψ_{11} is a wavelets Mexican Hat in consequent part of the rule that shown by following equation:

$$\Psi_l = \sum_{i=1}^{n} w_{il}\left(1-\left(\frac{x_j - b_{ij}}{c_{ij}}\right)^2\right)\exp\left(-\frac{1}{2}\left(\frac{x_j - b_{ij}}{c_{ij}}\right)^2\right) \tag{2}$$

The structure of two input one output FWNN model with two membership functions for each input is shown in Figure 1.

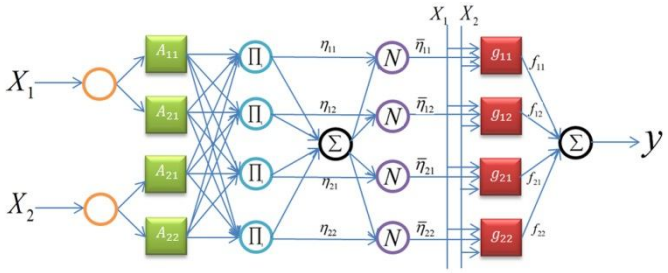

Figure II.1. The fuzzy wavelet neural network model Structure [10]

- **Layer 1:** This layer is the input layer that each nod in this layer transmits external input signals (x_1 and x_2) directly to the next layer [26].

- **Layer 2:** This layer is fuzzification layer. Neurons in this layer represent fuzzy sets used in the antecedents of fuzzy rules. The outputs of this layer are the values of the membership functions. The j^{th} Gaussian type membership function for the i^{th} input is given by:

$$A_{ij}(x_i) = \exp\left(-\frac{1}{2}\left(\frac{x_i-\mu_{ij}}{\sigma_{ij}}\right)^2\right) \qquad i=1,2\ and\ j=1,2 \tag{3}$$

- **Layer 3:** This layer is fuzzy rule layer. Each neuron in this layer represents activation strength of a fuzzy rule. The output of this layer is equal to multiplication of coming signals.

$$\eta_{ij} = A_{1j}(x_1).A_{2j}(x_2) \qquad i=1,2\ and\ j=1,2 \tag{4}$$

- **Layer 4:** This layer is normalization layer. Each neuron in this layer calculates the normalized activation strength of a given rule by [10]:

$$\bar{\eta}_{ij} = \frac{\eta_{ij}}{\sum_{i=1}^{2}\sum_{j=1}^{2}\eta_{ij}} \qquad i=1,2 \ \ and \ \ j=1,2 \tag{5}$$

- **Layer 5:** This layer is defuzzification layer. Each neuron in this layer receives initial inputs (x_1 and x_2) and the normalized activation strengths calculated previous layer as input and calculates the weighted consequent value of a given combination as follows:

$$f_l = \bar{\eta}_l \Psi_l \tag{6}$$

Where Ψ is:

$$\Psi_l = \sum_{i=1}^{n} w_{il}\left(1-\left(\frac{x_j - b_{ij}}{c_{ij}}\right)^2\right)\exp\left(-\frac{1}{2}\left(\frac{x_j - b_{ij}}{c_{ij}}\right)^2\right) \tag{7}$$

- **Layer 6:** This layer contains only a single node and it computes the overall output as the summation of all incoming signals, which is given by:

$$y = \sum_{l=1}^{m} f_l \tag{8}$$

II. Training algorithm

The FWNN training is to adjust a given function or input-output pairs by fine-tuning network parameters. Unknown parameters are center parameters (μ) and scaling parameters (σ) of Gaussian membership functions in first part of the rules, and translation (b), dilation (c) parameters of wavelet functions and weight (w) and bias (p) parameters in the resultant part of the rules[26].

The FWNN training is done by minimizing a performance indicator. For this aim, mean square error (MSE) is selected as performance indicator which is given by:

$$E = \frac{1}{N}\sum_{k=1}^{N}\left(y - y_d\right)^2 \tag{9}$$

Where N is the total number of input-output pairs of system that must be identifying.

Particle swarm optimization (PSO) introduced by Eberhart and Kennedy is a population based optimization algorithm just like Genetic algorithm (GA). PSO already has been applied successfully to the function optimization, image analysis, data clustering, neural network training etc.

Particle swarm optimization is a form of evolutionary computation algorithm based on the social metaphor of bird flocking or fish schooling. Like the genetic algorithm (GA), PSO is a population (swarm) based optimization tool. It has constructive cooperation between particles, and particles in the Swarm can share information. PSO has memory of the past, so knowledge of good solutions is retained by all particles. The application of PSO in neural network training involves creating a swarm of networks initialized with random weights. Each network is called a particle and is a candidate solution. The particles have the ability to retain their own best-ever state and communicate with each other. The swarm evolves in the search space by letting all the particles fly towards the best solutions they know of. In the local version of the PSO algorithm, each particle only communicates its neighbors; while in the global version, each particle can communicate with any other particles so everyone knows where the best-solution-so-far is. The whole training process with the global version PSO can be summarized as follows.

- Initialize the weight parameters of N networks (particles) with random numbers, where N is the number of particles in the swarm. Also initialize N velocity vectors with random numbers.
- Start the iteration by feeding each network with the training data, and calculating its mean squared error (MSE), which is used as the fitness criterion of the particles.
- Compare the MSE of each particle with its best history value, also called the personal best (pbest) MSE. If the current MSE is lower than the pbest MSE, update pbest MSE and store current weights as the pbest weights.
- Find the minimal newly calculated MSE in the swarm.
- Compare the minimal MSE with the global best (gbest) MSE. If the minimal MSE is lower than gbest MSE, update gbest MSE and store the corresponding weights as the gbest weights.

- Update the velocity and position vector of each particle according to equation (10) and (11):

$$V_i(t+1) = wV(t) + c_1 \times \rho_1 \times (W_{pbest} - W_i) + c_2 \times \rho_2 \times (W_{gbest} - W_i) \quad (10)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (11)$$

Where W is the inertia term, C_1 and C_2 are acceleration terms, ρ_1 and ρ_2 are uniformly distributed random numbers in [-1, 1], and W_{pbest}, W_{gbest}, and W_i are the weight vectors of pbest, gbest, and the current particle, respectively. The iteration loop continues until the MSE of the gbest is lower than the desired threshold or a maximum iteration number is reached. When the iteration is finished, the gbest weights are used as the training results.

It is reported in [20] that using the coefficient W, C_1, C_2 as following:

$$\varphi_1, \varphi_2 > 0 \quad (12)$$

$$\varphi = \varphi_1 + \varphi_2 > 4 \quad (13)$$

$$\chi = \frac{2}{\varphi - 2 + \sqrt{\varphi^2 - 4\varphi}} \quad (14)$$

$$W = \chi \quad (15)$$

$$C_1 = \chi \times \varphi_1 \quad (16)$$

$$C_2 = \chi \times \varphi_2 \quad (17)$$

the performance of PSO has been very good, and also if

$$\varphi_1 = \varphi_2 = 2.05 \quad (18)$$

it is optimal value for W, C_1 and C2 for better performance and accuracy and fast convergence.

III. Simulation examples

The FWNN structure and proposed training algorithm is applied to two system identification problems in order to show the performance of the training algorithm.

A. System identification case study 1

System identification involves finding the relation between the input and output of the system [1, 27]. In this example, the plant to be identified is given by following equation:

$$y(k) = 0.75y(k-1) + 0.025y(k-2)u(k-2) + 0.01u^2(k-3) + 0.02u(k-4) \quad (19)$$

The output of the system depends on two previous output values and three previous input values. However, only u (k-1) and y (k) are used as inputs to the FWNN models to predict y (k+1). Two membership functions are selected for each input of the FWNN model. In order to train the FWNN, 900 inputs are used similar to the inputs used in [28] and [29]and[10]. The half of the inputs is independent and identically distributed (i.i.d.) uniform sequence over [-2, 2] and the remaining is a sinusoid given by 1.05sin (πk/45). The FWNN is trained for 200 epochs that shows in Figure A.1. After training, the following input signal which is the same test signal as used in other methods is used for testing the performance of the proposed algorithm.

$$u(k) = \begin{cases} \sin(\pi k/25) & k < 250 \\ 1.0 & 250 < k < 500 \\ -1.0 & 500 \le k < 750 \\ 0.3\sin(\pi k/25) + 0.1\sin(\pi k/32) & \\ \quad + 0.6\sin(\pi k/10) & 750 \le k < 1000 \end{cases} \quad (20)$$

Root mean square error (RMSE) is taken as measure for system identification example that the value of RMSE is show in figure A.1. Figure A.2, shows the actual and predicted output of the plant for test data. From Table A.1,it

can be seen that the proposed algorithm illustrates much better performance than the models in [30, 31, 26] with less RMSE. The PSO algorithm gives better training results than the reported approaches.

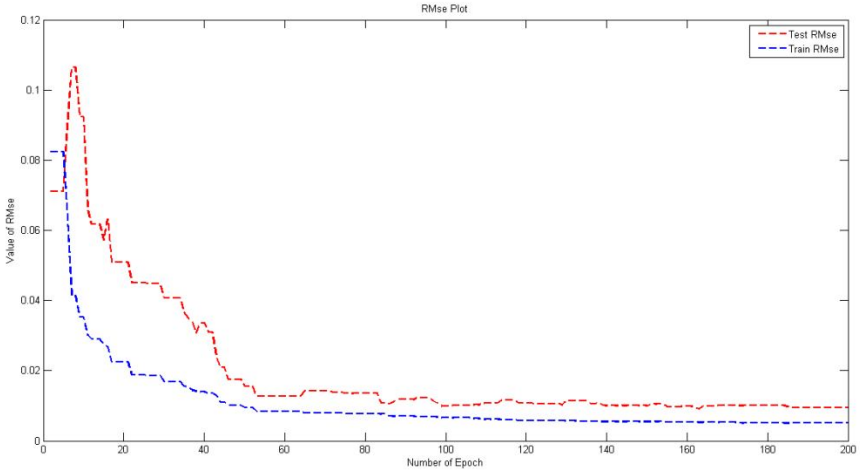

Figure A.1. RMSE values obtained during training and testing for system identification case study 1

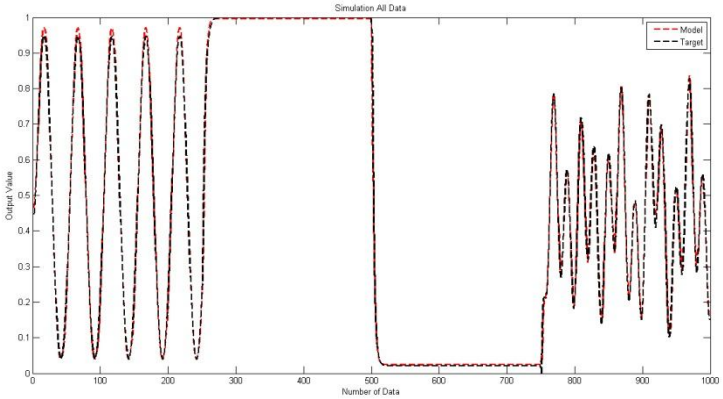

Figure A.2. Test results FWNN model with PSO training algorithm for system identification case study 1

Table A-1. Comparison of RFWNN model with other models for system identification case study 1

Models	Network Parameter	RMSE Training	RMSE Testing
ERNN [23]	54	0.036	0.078
RSONFIN [24]	49	0.03	0.06
TRFN-S [25]	33	0.0067	0.0313
FWNN [1]	27	0.01973	0.02260
FWNN [1]	43	0.01871	0.02016
RFWNN [17]	28	0.00968	0.02220
FWNN-PSO	32	0.00784	0.01573

B. System identification case study 2

This example considers the modeling of the nonlinear plant given by following equation:

$$y(k+1)=f\left(\left(y(k)\right),y(k-1),y(k-2),u(k),u(k-1)\right) \tag{21}$$

Where:

$$f\left(X_1,X_2,X_3,X_4,X_5,\right)=\frac{X_1X_2X_3X_4X_5\left(X_3-1\right)+X_4}{1+X_3^2X_2^2} \tag{22}$$

The same plant is also used in [5], [41], [57], [58], [59], [60] and [4].The current output of the plant depends on three previous output values and two previous input values. However, we use only y (k) and u (k) to predict y (k + 1) to compare our results with those of [5]. In order to train the FWNN models, 900 training inputs are generated as in the previous example. The number of MFs is also the same as in the previous example. To test the models for this plant, the input signal in (32) is used. The training and testing error values vs. the number of epochs are shown in Fig. 13.

The actual and predicted values for testing with PSO algorithm are shown in Fig B.2. As it is seen in Table B.1, are successful in identification than the compared models in the literature except the model proposed in [36].However, the model in [36] has 96 parameters whereas the proposed algorithm for FWNN in this paper has 32 parameters to be learned.

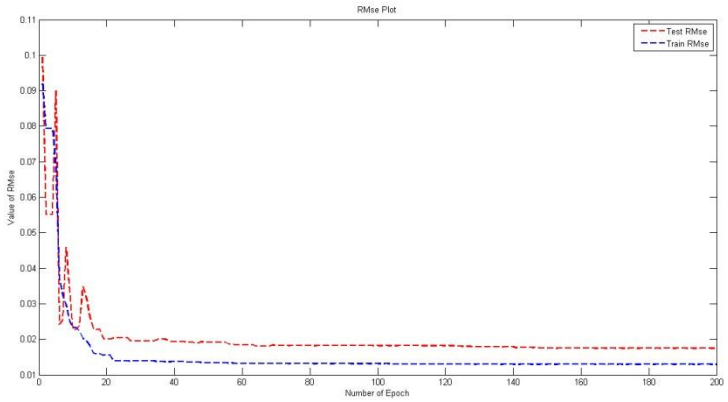

Figure B.1. RMSE values obtained during training and testing for system identification case study 2.

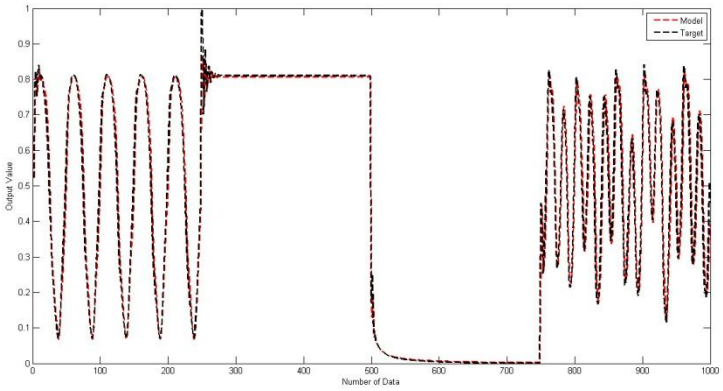

Figure B.2. Test results FWNN model with PSO training algorithm for system identification case study 2

Table B.1. COMPARISON OF FWNN MODELS WITH OTHER MODELS FOR SYSTEM IDENTIFICATION case study 2

Models	Network Parameter	RMSE Training	RMSE Testing
RFNN [26]	112	0.0114	0.0575
TRFN-S [27]	33	0.0084	0.0346
FWNN [1]	43	0.028232	0.030125
RFNN [28]	96	-----	0.0064
HRNFN [29]	21	-----	0.0493
FWNN-R [17]	28	0.015274	0.032116
FWNN-PSO	32	0.01288	0.01945

V. Conclusion

In this paper, the PSO algorithm is used to train a FWNN for the identification of nonlinear dynamic system. Simulation results demonstrated that, the accuracy of the PSO based training algorithm is better than the other reported training algorithmin the same iteration. In addition, our proposed method is faster than the existing gradient based approaches.

For future work the FWNN can be trained with PSO combine with an appropriate gradient based method such as LM.

REFERENCES

[1] R. H. Abiyev, and ,O. Kaynak, "Fuzzy Wavelet Neural Networks for Identification and Control of Dynamic Plants-A Novel Structure and Comparative Study.," *IEEE*, vol. 55, no. 8, pp. 3133 - 3140, Aug. 2008.

[2] K. S. Narendra. and , k.Parthasarathy, "Identification and control dynamical systems using neural networks," *IEEE Trans*, vol. 1, no. 1, p. 4–27, 1990.

[3] Oliver Nelles, Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models, Springer, December 2000.

[4] Sevcan Yılmaz , Yusuf Oysal , "A Fuzzy Wavelet Neural Network Model for System Identification," in *Ninth International Conference on Intelligent Systems Design and Applications*, 2009 .

[5] D.W.C. Ho, Ping-Au Zhang,Jinhua Xu, "Fuzzy wavelet networks for function learning," *IEEE Trans.Fuzzy Syst*, vol. 6, no. 6, pp. 200-211, Feb. 2001.

[6] J. Zhang, G.G. Walter, and W.N.W. Lee, "Wavelet neural networks for function learning," *IEEE Trans.Signal Process*, vol. 43, no. 6, pp. 1485-1497, June 1995.

[7] Y. Chen, B. Yang, and J. Dong, "Time-series prediction using a local linear wavelet neural network," *Neurocomputing*, vol. 69, pp. 449-465, 2006.

[8] Q. Zhang and A. Benveniste, "Wavelet networks," *IEEE Trans. Neural network*, vol. 3, no. 6, p. 889–898, Nov. 1992.

[9] J. Zhang, G. G. Walter, Y. Miao, and W. N. W. Lee, "Wavelet neural networks for function learning," *IEEE Trans,Signal Process*, vol. 43, no. 6, p. 1485–1497, Jun. 1995.

[10] P. Wcrhos, "Backpropagation through limc: what it docs and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. I550 -1560, oct,1990.

[11] M.T. Hagan. H.B. Demulh and M. Bcale, " Neural Network Design," *PWS Publishing Company*, 1996.

[12] R. .. Batlit, "first and second order methods of learning:between the gradient desent and newoton's metod," *neural computation*, vol. 4, no. 2, pp. 141-66, 1991.

[13] M.T. Hagan and M.B. Menhaj, "Training feedfoward networks with the Marquardt algorithm," *IEEE 'Trans. on Neural Networks*, vol. 5, no. 6, pp. 989-993, nov-1994.

[14] Venu G. Gudise and Ganesh K. Venayagamoorthy, "Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks," *Senior- Member IEEE* .

[15] Kennedy ,J., Eberhart , R.C., and Shi ,Y., Swarm intelligence, Morgan Kaufmann Publishers, 2001.

[16] Ge Hongwei, Liang Yanchun, "Identification for non-linear systems based on particle swarm optimization and recurrent neural network," vol. 2, pp. 27-30, May 2005.

[17] Sevcan Yılmaz,Yusuf Oysal, "A Fuzzy Wavelet Neural Network Model for System Identification," in *Intelligent Systems Design and Applications*, 2009.

[18] K. S. Narendra and K. Parthasarathy, "Identification and control dynamical systems using neural networks," *IEEE Trans.Neural Netw*, vol. 1, no. 1, p. 4–27, 1990.

[19] C. F. Juang and C. T. Lin, "A recurrent self-organizing neural fuzzy inference network," *IEEE Trans. Neural Netw*, vol. 10, no. 4, p. 828–845, 1999.

[20] C. F. Juang, "A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms," *IEEE Trans. Fuzzy Syst,* vol. 10, no. 2, p. 155–170, 2002.

[21] C. F. Juang and C. T. Lin, "A recurrent self-organizing neural fuzzy inference network," *IEEE Trans. Neural Netw,* vol. 10, no. 4, p. 828–845, 1999.

[22] J. L. Elman, "Finding structure in time," *Cognit. Sci.,* vol. 14, p. 179–211, 1990.

[23] J. L. Elman, "Finding structure in time," *Cognit. Sci,* vol. 14, p. 179–211, 1990.

[24] C. F. Juang and C. T. Lin, "A recurrent self-organizing neural fuzzy inference network," *IEEE Trans. Neural Netw,* vol. 10, no. 4, p. 828–845, 1999.

[25] C. F. Juang, "A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms," *IEEE Trans. Fuzzy Syst,* vol. 10, no. 2, p. 155–170, 2002.

[26] C.-H. Lee and C.-C. Teng, "Identification and control of dynamic systems using recurrent fuzzy neural networks," *IEEE Trans. Fuzzy Syst.,* vol. 8, no. 4, p. 349–366, Aug. 2000.

[27] C.-F. Juang, "A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms," *IEEE Trans Fuzzy Syst,* vol. 10, no. 2, p. 155–170, Apr. 2002.

[28] A. Hussain, "A new neural network structure for temporal signal processing," *Proc. IEEE ICASSP,* vol. 4, p. 3341–3344, 1997.

[29] J.-S. Wang and Y.-P. Chen, "A Hammerstein recurrent neurofuzzy network with an Online minimal realization learning algorithm," *IEEE Trans. Fuzzy Syst,* vol. 16, no. 6, p. 1597–1612, Dec. 2008.