Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/091L16

A Self-Organizing Recurrent Wavelet Neural Network for Nonlinear Dynamic System Identification

Cheng-Jian Lin[1,*], *Chun-Cheng Peng*[1], *Cheng-Hung Chen*[2] *and Hsueh-Yi Lin*[1]

[1] Dept. Computer Science and Information Engineering, National Chi-Yi University of Technology, Taichung City, Taiwan, R. O. C.
[2] Dept. Electrical Engineering, National Formosa University, Yunlin County, Taiwan, R. O. C.

Received: 17 Nov. 2013, Revised: 18 Mar. 2014, Accepted: 19 Mar. 2014
Published online: 1 Feb. 2015

Abstract: To solve identification of nonlinear dynamic systems, a recurrent wavelet neural network (RWNN) model is proposed in this paper. The proposed RWNN model has four-layer structure. Temporal relations embedded in the network by adding some feedback connections representing the memory units in the second layer. An online learning algorithm, which consists of structure learning and parameter learning, is proposed and is able to construct the wavelet neural network dynamically. The structure learning is based on the input partitions to determine the number of wavelet bases, and the parameter learning is based on the supervised gradient descent method to adjust the shape of wavelet functions, feedback weights, and the connection weights. Computer simulations were conducted to illustrate the performance and applicability of the proposed model.

Keywords: Recurrent neural network, wavelet bases, identification, online learning, backpropagation, degree measure.

1 Introduction

Artificial neural networks are powerful empirical modeling tools that can be trained to represent complex multi-input multi-output nonlinear systems. There are two major classes of neural networks that have become more important in recent years, namely, feedforward neural networks (NN) [1]-[3] and recurrent neural networks (RNN) [4]-[7]. It is known that a three-layer feedforward NN is capable of approximating any continuous map arbitrarily closely. Since for a dynamic system [1], the output is a function of past output or past input or both, identification and control of this system is not as straightforward as a static system. Due to the feedforward NN is a static mapping and is without the aid of tapped delays. The feedforward NN is unable to represent a dynamic system mapping. On the other hand, the RNN has superior capabilities than NN, such as dynamic and the ability to store information for later use. Therefore, the RNN have the vantage of dealing with temporal problems, which have been found to be difficult for feedforward NN. Regardless of their type, however, neural networks are generally disadvantaged by their ”black box” format, and lack a systematic way to determine the appropriate model structure, have no localizability, and converge slowly. A suitable local approximation approach is proposed to overcoming the disadvantages of global approximation networks. That is, the global activation function is substituted by localized basis functions. For the local approximation method, only a small subset of the network parameters is engaged at each point in the input space. The network transparency may be improved by adopting the wavelet decomposition technique from the field of adaptive signal processing. Due to the local properties of wavelets, arbitrary functions can be approximated by the truncated discrete wavelet transform.

Recently, many researches proposed wavelet neural networks for identification and control [8]-[14]. Ikonomopoulos and Endou [8] proposed the analytical ability of the discrete wavelet decomposition with the computational power of radial basis function networks. Members of a wavelet family were chosen through a statistical selection criterion that constructs the structure of the network. Ho et al. [9] used the orthogonal least squares (OLS) algorithm to purify the wavelets from their candidates, which avoided using more wavelets than required and often resulted in an overfitting of the data and a poor situation in [10]. Lin et al. [11] proposed a wavelet neural network to control the moving table of a

* Corresponding author e-mail: cjlin@ncut.edu.tw

linear ultrasonic motor (LUSM) drive system. They chose an initialization for the mother wavelet based on the input domains defined by the examples of the training sequence. Huang and Huang [12] proposed an evolutionary algorithm for optimally adjusted wavelet networks. However, the selections of wavelet bases were based on practical experimentation or trial-and-error tests.

The objective of this paper is to introduce a recurrent wavelet neural network (RWNN) model with online learning algorithm. The architecture of RWNN model enables them to preserve past states of the networks. Therefore, the RWNN model has the capability to deal with temporal problems. An online learning algorithm, which consists of structure learning and parameter learning, is proposed and is able to construct the wavelet neural network dynamically. The structure learning is based on the input partitions to determine the number of wavelet bases, and the parameter learning is based on the supervised gradient descent method to adjust the shape of wavelet functions, feedback weights, and the connection weights. In the initial form, there are no wavelet bases and wavelet functions. They are created and begin to grow as the first training input data arrives. Thus, the user need not be given any a priori knowledge or initial information on the wavelet bases and functions. Finally, the RWNN model is applied in several identification problems. The advantages of the proposed RWNN model are summarized as follows: 1) it converges quickly; 2) it is constructed automatically; 3) it requires much lower adjustable parameters; and 4) it has much lower rms error.

The rest of this paper is organized as follows. Section 2 presents the proposed structure of the RWNN, while the online learning algorithm is exhibited in section 3. After the illustrative example is shown in section 4, section 5 concludes this paper.

2 Structure of the Recurrent Wavelet Neural Network

In the *static* wavelet-base neural network (WNN), the input data in the input layer of the network at time s is $x(s) = [x_1(s), x_2(s), \ldots, x_i(s), \ldots, x_n(s)]^T$, where T is the transpose and n is the number of dimensions. Noted that in ordinary wavelet neural network model applications, it is often useful to normalize the input vectors $x(s)$ into the interval [0, 1]. Then, the activation functions of the wavelet nodes in the wavelet layer are derived from the mother wavelet $\phi(x(s))$, with a dilation of d and a translation of t [10]:

$$\phi_{d.t}(x(s)) = 2^{d/2}\phi(2^d x(s) - t). \quad (1)$$

The mother wavelet is selected so that it constitutes an orthonormal basis in $L^2(\Re^n)$. The derivation of a differentiable *Mexican-hat* function is adopted as a mother wavelet herein,

$$\phi(x(s)) = (1 - \|x(s)\|^2)e^{-\|x(s)\|^2/2}, \quad (2)$$

where $\|x(s)\|^2 = x^T x$. Therefore, the activation function of the jth wavelet node connected with the ith input data is represented as:

$$\phi_{d_j.t_j}(x_i(s)) = 2^{d_{ij}/2}(1 - \|2^{d_{ij}}x_i(s) - t_{ij}\|^2)e^{-\|2^{d_{ij}}x_i(s)-t_{ij}\|^2/2}, \quad (3)$$

where $i = 1, \ldots, n$, $j = 1, \ldots, m$, n is the number of input-dimensions, and m is the number of the wavelets. The wavelet functions of Eq. (3) with various dilations and translations are presented in Fig. 1. Equation (3) indicated that the enforcement of *static mapping*.

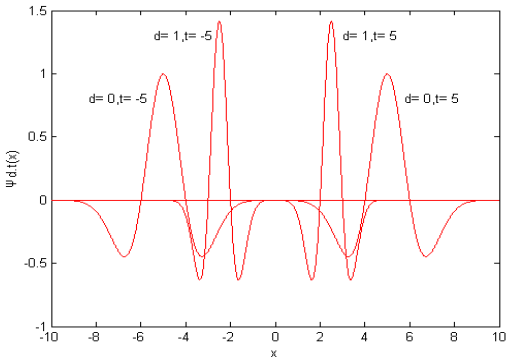

Fig. 1: Wavelet bases with various dilations and translations.

The structure of the RWNN model is shown in Fig. 2. The proposed RWNN model is designed as a four-layer structure, which is comprised of an input layer, wavelet layer, product layer, and output layer. Temporal relations embedded in the network by adding some feedback connections representing the memory units in the second layer. In RWNN model, to store information for later use, the *dynamic mapping* is adopted as follows:

$$\phi_{d_j.t_j}(z_{ij}^{-1}(s)) = 2^{d_{ij}/2}(1 - \|2^{d_{ij}}z_{ij}^{-1}(s) - t_{ij}\|^2)e^{-\|2^{d_{ij}}z_{ij}^{-1}(s)-t_{ij}\|^2/2}, \quad (4)$$

where

$$z_{ij}^{-1}(s) = x_i(s) + \theta_{ij} \cdot \phi_{d_j.t_j}(z_{ij}^{-1}(s-1)), \quad (5)$$

θ_{ij} is the feedback weight, $x_i(s)$ is the ith input variable at time s, $i = 1, \ldots, n$, and $j = 1, \ldots, m$. It is clear that the input of this layer contains the memory terms $z_{ij}^{-1}(s-1)$, which store the past information of the network. Then, each wavelet in the product layer is labeled $\prod$, i.e., the product of the jth multi-dimensional wavelet with $Z_{ij}^{-1}(s) = [z_{1j}^{-1}, \ldots, z_{nj}^{-1}]$ can be defined as

$$\psi_j(Z_{ij}^{-1}(s)) = \prod_{i=1}^{n} \phi_{d_j.t_j}(z_{ij}^{-1}(s)). \quad (6)$$

According to the theory of multi-resolution analysis (MRA) [10,14], any $f \in L^2(\Re)$ can be regarded as a

linear combination of wavelets at different resolution levels. For this reason, the function f is expressed as

$$Y_l(s) = f \approx \sum_{j=1}^{m} w_j^l \psi_j(Z_j^{-1}(s)), \tag{7}$$

where $Y_l(s) = [Y_1(s), \ldots, Y_p(s)]$ means the lth output of the RWNN model at time s. If $\psi_j = [\psi_1, \ldots, \psi_m]$ is used as a nonlinear transformation function of hidden nodes and weight vectors and $w_j^l = [w_1^l, \ldots, w_m^l]$ defines the connection weights, then Eq. (7) can be considered the functional expression of the RWNN modeling function $Y_l(s)$.

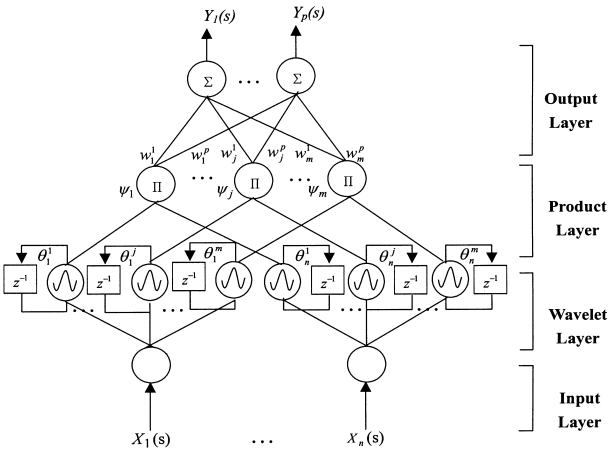

Fig. 2: The architecture of the RWNN model.

3 An On-line Learning Algorithm

In this section, the degree measure method and the well-known back propagation (BP) algorithm are used concurrently for constructing and adjusting the RWNN controller. The degree measure method is used to decide the number of wavelet bases in the wavelet layer and the product layer. On the other hand, the BP algorithm is used to adjust the parameters of the wavelet bases, feedback weights, and connection weights. The details of the algorithm are presented below.

3.1 The Structure Learning Scheme

Initially, there are no wavelet bases in the RWNN model. The first task is to decide when a new wavelet base is generated. For each incoming pattern $x_i(s)$, the firing strength of a wavelet base can be regarded as the degree of the incoming pattern belonging to the corresponding wavelet base. An input datum $x_i(s)$ with a higher firing strength means that its spatial location is nearer to the center of the wavelet base t_{ij} than those with smaller firing strength. Based on this concept, the firing strength obtained from Eq. (6) in the product layer can be used as the degree measure

$$F_j = |\psi_j|, \tag{8}$$

where $j = 1, \ldots, q$, q is the number of existing wavelet bases, and $|\psi_j|$ is the absolute value of ψ_j. According to the degree measure, the criterion of a new wavelet base generated for new incoming data is described as follows:

Find the maximum degree $F_{\max}$

$$F_{\max} = \max_{1 \le j \le q} F_j. \tag{9}$$

If $F_{\max} \le \overline{F}$, then a new wavelet base is generated, where $\overline{F}$ is a pre-specified threshold that should decay during the learning process, limiting the size of the RWNN model.

$$t_{q+1} = x_i(s), \tag{10}$$

$$d_{q+1} = 0, \tag{11}$$

and

$$w_{q+1} = \theta_{q+1} = \text{random value}, \tag{12}$$

where $x_i(s)$ is the new incoming data at time s; the connection weight w_{q+1} of the output layer and feedback weight θ_{q+1} are selected from the range between -1 and 1 randomly; and the dilation d_{q+1} is set to zero to obtain a higher firing strength for the input value $x_i(s)$ (see Fig. 2).

The concise online degree measure method of the RWNN model is shown as follows:

Initialization;
do{
  IF $x_i(s)$ *is the first incoming pattern, do*{
    Generate a new wavelet base;
    with translation $t_{q+1} = x_i(s)$*;*
    dilation $d_{q+1} = 0$*;*
    connection weight $w_{q+1} \in [-1, 1]$*;*
    feedback weight $\theta_{q+1} \in [-1, 1]$*;*
  }
  ELSE for each newly incoming pattern, do{
    Executing the degree measure method;
    IF $F_{\max} \le \overline{F}$*, do*{
      Generate a new wavelet base;
      with translation $t_{q+1} = x_i(s)$*;*
      dilation $d_{q+1} = 0$*;*
      connection weight $w_{q+1} \in [-1, 1]$*;*
      feedback weight $\theta_{q+1} \in [-1, 1]$*;*
    }
  }
} *until the task is finished.*

3.2 The Parameter Learning Scheme

After the network structure has been adjusted according to the current training pattern, the network then enters the second learning step to adjust the parameters of the wavelet base, feedback weight and the connection weight (t, d, θ, and w) with the same training pattern. The parameter-learning algorithm is based on a set of MIMO pairs $\{x_i(s), Y_l^{des}(s)\}$. If the lth error function e_l is defined

$$e_l = (Y_l(s) - Y_l^{des}(s)), \tag{13}$$

where $Y_l(s)$ is the lth model output and $Y_l^{des}(s)$ is the lth desired output at time s, then the cost function E can be defined as

$$E = \frac{1}{2p}\sum_{l=1}^{p} e_l^2 \tag{14}$$

and can be minimized by all adjustable parameters using an iterative computational scheme.

Assuming that W is the adjustable parameter in the wavelet layer and the output layer, the general learning rule used is

$$W(s+1) = W(s) + \Delta W = W(s) + \eta(\frac{\partial E}{\partial W}), \tag{15}$$

where η and s represent the learning rate and the iteration number, respectively. The gradient of the cost function E in Eq. (14) with respect to the vector of arbitrarily adjustable parameter W is defined as

$$\frac{\partial E}{\partial W} = -\frac{1}{p}\sum_{l=1}^{p} e_l \frac{\partial Y_l}{\partial W}. \tag{16}$$

With the above equation defined, we can infer that the free parameters adjusted in the RWNN is as follows:
The connection weight of the output layer is updated by

$$w_j^l(s+1) = w_j^l(s) + \Delta w_j^l, \tag{17}$$

where

$$\Delta w_j^l = -\eta_w \frac{\partial E}{\partial w_j^l} = -\eta_w \cdot \frac{1}{p} \cdot e \cdot \psi_j. \tag{18}$$

Similarly, the updated laws of t_{ij}, d_{ij}, and θ_{ij} are shown as follows:

$$t_{ij}(s+1) = t_{ij}(s) + \Delta t_{ij}, \tag{19}$$

$$d_{ij}(s+1) = d_{ij}(s) + \Delta d_{ij}, \tag{20}$$

and

$$\theta_{ij}(s+1) = \theta_{ij}(s) + \Delta \theta_{ij}, \tag{21}$$

where

$$\begin{aligned}\Delta t_{ij} &= -\eta_t \frac{\partial E}{\partial t_{ij}} \\ &= -\eta_t \cdot \frac{1}{p}\sum_{l=1}^{p} e_l \cdot w_j^l \cdot \psi_j \cdot \left\{ \frac{2^{d_{ij}} z_{ij}^{-1}(s) - t_{ij}}{1-(2^{d_{ij}} z_{ij}^{-1}(s) - t_{ij})^2} \right. \\ &\left. \cdot \left[2 + \left(1 - (2^{d_{ij}} z_{ij}^{-1}(s) - t_{ij})^2\right)\right] \right\},\end{aligned} \tag{22}$$

$$\begin{aligned}\Delta d_{ij} &= -\eta_d \frac{\partial E}{\partial d_{ij}} = -\eta_d \cdot \frac{1}{p}\sum_{l=1}^{p} e_l \cdot w_j^l \cdot \psi_j \\ &\cdot \left\{ \frac{ln2}{2} - 2^{d_{ij}} \cdot z_{ij}^{-1}(s) \cdot ln2 \cdot (2^{d_{ij}} z_{ij}^{-1}(s) - t_{ij}) \right. \\ &\left. \cdot \left[1 + \frac{2}{1-(2^{d_{ij}} z_{ij}^{-1}(s) - t_{ij})^2} \right] \right\},\end{aligned} \tag{23}$$

$$\begin{aligned}\Delta \theta_{ij} &= -\eta_\theta \frac{\partial E}{\partial \theta_{ij}} = -\eta_\theta \cdot \frac{1}{p}\sum_{l=1}^{p} e_l \cdot w_j^l \cdot \psi_j \\ &\cdot \left\{ \frac{2^{d_{ij}} z_{ij}^{-1}(s) - t_{ij}}{1-(2^{d_{ij}} z_{ij}^{-1}(s) - t_{ij})^2} \cdot \phi_{d_j . t_j}(z_{ij}^{-1}(s-1)) \right. \\ &\left. \cdot 2^{d_{ij}} \cdot \left((2^{d_{ij}} z_{ij}^{-1}(s) - t_{ij})^2 - 3 \right) \right\},\end{aligned} \tag{24}$$

4 Illustrative Examples

To certify the performance of the RWNN model for temporal problems, several examples and performance contrasts with some other recurrent networks are presented in this section. These parameters (η_t, η_d, η_θ, η_w, d_{init}, $\overline{F}$) are set in advance, and the number of training epochs for the RWNN model in each example is determined based on the desired accuracy.

Example 1: Prediction of Time Sequence

To clearly verify if the proposed RWNN model can learn the temporal relationship, a simple time sequence prediction problem found in [8] is used for test in the following example.

The test bed used is shown in Fig. 3(a). This is an "8" shape made up of a series with 12 points which are to be presented to the network in the order as shown. The RWNN model is asked to predict the succeeding point for every presented point. Obviously, a static network cannot accomplish this task, since the point at coordinate (0, 0) has two successors: point 5 and point 11. The RWNN model must decide the successor of (0, 0) based on its predecessor; if the predecessor is 3, then the successor is 5, whereas if the predecessor is 9, the successor is 11.

In this example, the RWNN model contains only two input nodes, which are activated with the two dimensional coordinate of the current point, and two output nodes, which represent the two dimensional coordinate of the predicted point. The initial parameters are set as $\eta_t = \eta_d = \eta_\theta = \eta_w = 0.05$, $d_{init} = 0$, and $\overline{F} = 0.6$. The training process is continued for 1000 epochs. Starting at zero, the number of wavelet nodes grows dynamically for incoming training data. The final root-mean-square (rms) error of 0.000014 is achieved. The free parameters are obtained at end of learning as follows:

$$d_{ij} = \begin{bmatrix} -0.284 & 0.118 & 0.019 & -0.11 & 0.16 & -0.143 & -0.046 & 0.002 & -0.01 & -0.1 & -0.01 & 0.02 \\ 0.435 & 0.211 & -0.121 & 0.599 & -0.215 & 0.405 & 0.525 & -0.009 & 0.144 & 0.02 & 0.003 & 0.06 \end{bmatrix},$$

$$t_{ij} = \begin{bmatrix} -0.182 & 0.59 & 0.943 & 0.099 & -0.943 & -0.77 & 0.242 & 0.49 & 0.917 & -1.165 & -0.0009 & -0.346 \\ 0.532 & 1.188 & 0.529 & -0.095 & -0.706 & -0.779 & -1.05 & -1.04 & -0.175 & 0.451 & 1.001 & -0.982 \end{bmatrix},$$

$$w_{1j} = \begin{bmatrix} -0.274 & 0.726 & 0.388 & -0.965 & -0.649 & -0.349 & 0.578 & 0.269 & -0.113 & -0.086 & -0.056 & 0.0166 \\ 0.787 & 0.189 & 0.09 & 0.017 & -0.342 & -0.765 & -0.785 & -0.01 & -0.221 & 0.209 & -0.199 & 0.527 \end{bmatrix},$$

and

$$\theta_{ij} = \begin{bmatrix} 0.247 & 0.111 & -0.399 & -0.328 & -0.464 & 0.197 & 0.263 & -0.306 & -0.04 & -0.389 & -0.126 & 0.3 \\ -0.598 & -0.118 & 0.003 & 0.172 & -0.347 & 0.233 & 0.52 & 0.105 & -0.521 & 0.111 & 0.099 & -0.525 \end{bmatrix}.$$

Figure 3(b) shows the prediction results by the trained RWNN model. Simulation results show that we can obtain perfect prediction capability.

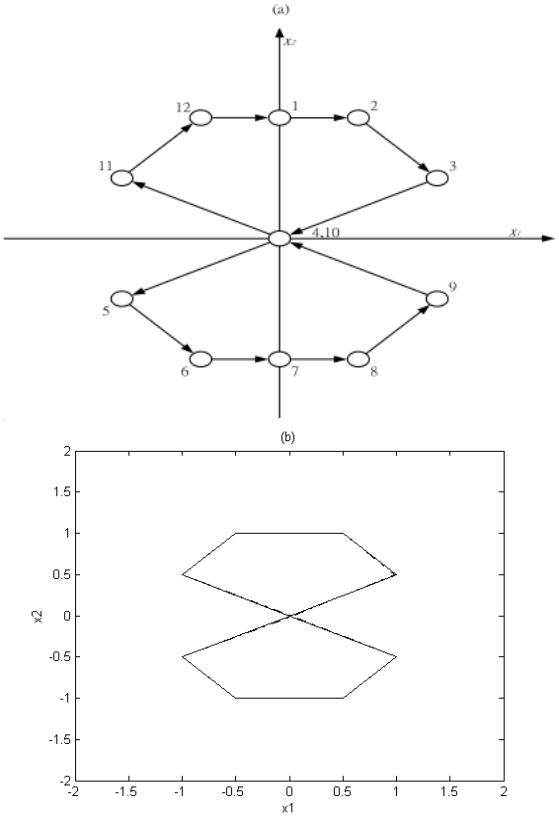

Fig. 3: Simulation results of time sequence prediction. (a) Test bed for the sample prediction experiment in Example 1. (b) Results of prediction using the RWNN model after 1000 training epochs.

Recently, Lee and Teng [15] proposed a model, called recurrent fuzzy neural network (RFNN) architecture, for learning and tuning a fuzzy predictor. They adopted standard zero-order TSK-type fuzzy model. For initializing parameters of the RFNN model, the rule number should be given in advance. But, the users need not give it any *a priori* knowledge or even any initial information for our proposed model. The RFNN model [15] and a traditional (non-recurrent) fuzzy neural network (FNN) [16] are also applied to this time prediction problem. Figure 4(a) shows the prediction results using the RFNN model. In this figure, the RFNN also obtain prediction capability, but some time prediction points cannot be matched exactly. Figure 4(b) shows that a feedforward FNN model cannot predict successfully, because of its static mapping. Figure 4(c) shows the learning curves of the RWNN model, the RFNN model [15] and the FNN model [16]. The learning curve of the RWNN appears to have an oscillation at the beginning of learning. This situation reflects the structure changing in the early stage of learning. In this figure, our model converges quickly and obtains a small rms error. To give a clear understanding of this performance comparison with the RFNN model and the FNN model on the same problem is made in Table 1. Computer simulations were conducted to illustrate the performance and applicability of the proposed model.

Table 1: Performance comparison of various existing models on time sequence prediction, with 1000 epochs.

	Rules/nodes	Parameters	RMS error
RWNN	**12**	**96**	**0.000014**
RFNN [15]	12	96	0.0072
FNN [16]	12	108	0.148

Example 2: Identification of Nonlinear Dynamic System

Consider the following dynamic plant with longer input delays:

$$y_p(s+1) = 0.72y_p(s) + 0.025y_p(s-1)u(s-1) + 0.01u^2(s-2) + 0.2u(s-3) \quad (25)$$

This plant is the same as that used in Kim et al. [17]. In our model, only with two input values, $y_p(s)$ and $u(s)$, are fed to the RWNN model to determine the output $y_p(s)$. The training input are independent and identically distributed (i.i.d.) uniform sequence over $[-2, 2]$ for about half of the training time and a single sinusoid signal given by $1.05\sin(\pi s/45)$ for the remaining training time. There is no repetition on these 900 training data, i.e., different training sets for each epoch. The checking input signal $u(s)$ as the following equation is used to determine the identification results

$$u(s) = \begin{cases} \sin\left(\frac{\pi \cdot s}{25}\right), & 0 < s < 250 \\ 1.0, & 250 \le s < 500 \\ -1.0, & 500 \le s < 750 \\ 0.3\sin\left(\frac{\pi \cdot s}{25}\right) + 0.1\sin\left(\frac{\pi \cdot s}{32}\right) & \\ \quad +0.6\sin\left(\frac{\pi \cdot s}{10}\right), & 750 \le s < 1000 \end{cases} \quad (26)$$

During the training, only 10 epochs be used, where are 900 time steps in each epoch. The initial parameters are set as $\eta_t = \eta_d = \eta_\theta = \eta_w = 0.05$, $d_{init} = 0$, and $\overline{F} = 0.07$. After training, the final rms error is 0.00028, and two wavelet nodes are generated. These obtained free

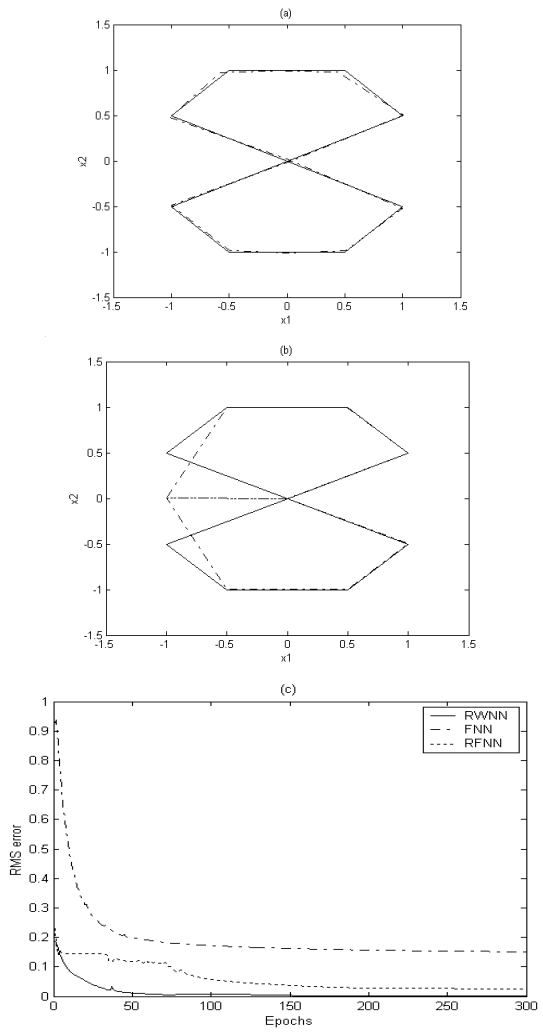

Fig. 4: (a) Results of prediction using the RFNN [15] after 1000 training epochs. (b) Results of prediction using the FNN [16] after 1000 training epochs. (c) Learning curves of the RWNN model, the RFNN model [15] and the FNN model [16].

parameters are

$$d_{ij} = \begin{bmatrix} 0.437 & 0.446 \\ -0.862 & -0.455 \end{bmatrix}, \; t_{ij} = \begin{bmatrix} 0.946 & -0.9 \\ 0.535 & -0.628 \end{bmatrix},$$

$$\theta_{ij} = \begin{bmatrix} -0.281 & 0.329 \\ -0.44 & 0.465 \end{bmatrix}, \quad w_{lj} = [1.22 \; -0.957].$$

Figure 5(a) shows the outputs of the plant and the RWNN model. The results show the perfect identification capability of the RWNN model. Figure 5(b) illustrates the error between the desired output and the RWNN output. The learning curves of the WRFNN model and the RFNN model [15] are shown in Fig. 5(c). In this figure, we also obtain a smaller rms error and converge quickly than the RFNN model [15]. Finally, we compare the performance of our model with that of other existing recurrent methods (RFNN [15], ERNN [18], RSONFIN [19], and TRFN-S [20]). The comparison results are tabulated in Table 2. As shown in Table 2, the numbers of adjustable parameters and rms error in our model are rather smaller than other recurrent methods under the same training epochs.

Table 2: Performance comparison of various recurrent methods on the identification problem, with 10 epochs.

	Parameters	RMS err. (train)	RMS err. (test)
RWNN	**14**	**0.00028**	**0.0012**
TRFN-S [20]	33	0.0067	0.0313
RFNN [15]	21	0.00181	0.00402
RSONFIN [19]	49	0.03	0.06

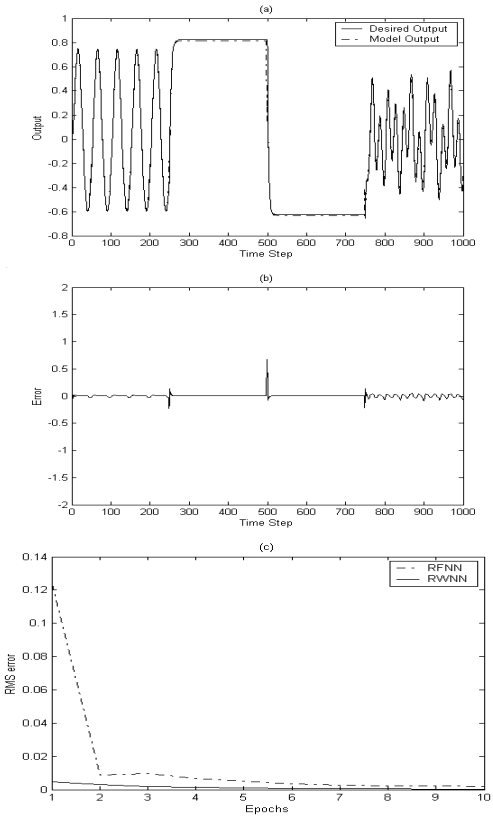

Fig. 5: Simulation results of the RWNN model for dynamic system identification in Example2. (a) The outputs of the plant and the RWNN. (b) The error between the RWNN output and the desired output. (c) Learning curves of the RWNN model and the RFNN model [15].

Example 3: Identification of Chaotic System

The discrete time Henon system is repeatedly used in the study of chaotic dynamics and is not exceedingly

simple in the sense that it is of second order with one delay and two parameters [21]. This chaotic system is described by

$$y(s+1) = -H \cdot y^2(s) + Q \cdot y(s-1) + 1.0, \qquad (27)$$

for $s = 1, 2, \ldots$, which, with H=1.4 and Q=0.3, produces a chaotic strange attractor as shown in Fig. 6(a). For this training, the input of the RWNN model is $y(s-1)$ and the output is $y(s)$. Now, the training input patterns sampled randomly (1000 pairs) from system over the interval $y(s-1) \in [-1.5, 1.5]$. Then, the RWNN model is used to approximate the chaotic system.

In applying the RWNN model to this example, only 100 epochs are used. Here the initial point is $[y(1), y(0)]^T = [0.4, 0.4]^T$. The initial parameters are set as $\eta_t = \eta_d = \eta_\theta = \eta_w = 0.05$, $d_{init} = 0$, and $\overline{F} = 0.36$. After training, there are six wavelet nodes are generated. The obtained free parameters are:

$$d_{ij} = [-0.211\ -0.226\ -1.936\ -1.996\ -0.05\ 0.186],$$

$$t_{ij} = [-0.321\ -0.453\ -0.72\ 0.494\ 2.184\ 1.06],$$

$$\theta_{ij} = [0.081\ -0.083\ -1.556\ -2.24\ 0.022\ -0.0189],$$

and

$$w_{lj} = [1.592\ 1.551\ -4.555\ -2.196\ -1.408\ 1.164].$$

The phase plane of this chaotic system after training for the FNN model [16] and the RWNN are shown in Fig. 6(b) and Fig. 6(c). In Fig. 6(b), the FNN model is inappropriate for chaotic dynamics system because of its static mapping. To give a clear understanding of this performance comparison with the RFNN model and the FNN model on the same problem is made in Table 3. The proposed RWNN model needs fewer wavelet nodes and obtains a smaller rms error than the RFNN model and the FNN model.

Table 3: Performance comparison of various methods in this chaotic system in Example 3, with 100 epochs.

	Rules/nodes	Parameters	RMS err. (train)	RMS err. (test)
RWNN	**6**	**24**	**0.0017**	**0.0017**
RFNN [15]	8	32	0.0141	0.0145
FNN [16]	8	24	0.1338	0.1557

5 Conclusion

In this paper, we propose a recurrent wavelet neural network (RWNN) model for solving temporal problems. We adopt the orthogonal function as wavelet neural network bases. Due to its multiscale, multiresolution, and localization, the RWNN can accurately capture the nonlinear behavior of systems. Adding feedback connections in the second layer, where the feedback units act as memory elements, develop the temporal relations embedded in the RWNN. An online learning algorithm is proposed to construct model and tune parameters automatically. The experimental results strongly demonstrate that the learning scheme is very effective for identification of dynamic systems.

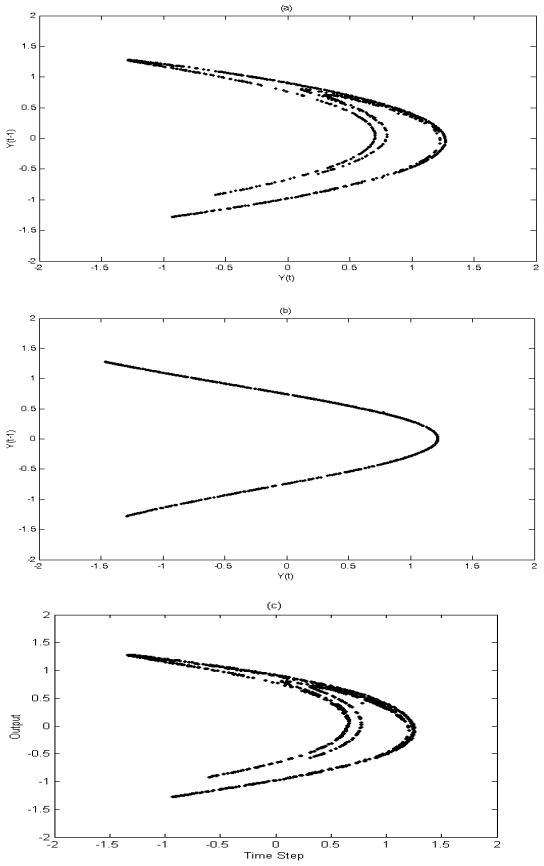

Fig. 6: (a) Check data of this chaotic system. (b) Result of identification using the FNN model [16] for the chaotic system. (c) Result of identification using the RWNN model for the chaotic system.

Acknowledgement

This work is supported by National Science Council (NSC) (NSC102-2221-E-167-023).

References

[1] K. S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Transactions on Neural Networks **1**, 4-27 (1990).

[2] M. Khalid and S. Omatu, A neural network controller for a temperature control system, IEEE Transactions on Control Systems **12**, 58-64 (1992).
[3] J. D. Johnson, Neural networks for control, Neurocomputing **14**, 301-302 (1997).
[4] F. J. Lin, C. H. Lin, C. H. Hong, Robust control of linear synchronous motor servo drive using disturbance observer and recurrent neural network compensator, Electric Power Applications **147**, 263-272 (2000).
[5] C. C. Ku and K. Y. Lee, Diagonal recurrent neural networks for dynamic systems control, IEEE Transactions on Neural Networks **6**, 144-156 (1995).
[6] A. Karakasoglu, S. I. Sudharsanan, M. K. Sundareshan, Identification and decentralized adaptive control using dynamical neural networks with application to robotic manipulators, IEEE Transactions on Neural Networks **4**, 919-930 (1993).
[7] L. Jin, P. N. Nikiforuk, M. M. Gupta, Dynamic recurrent neural networks for control of unknown nonlinear systems, J. Dyn. Syst., Measur., Contr. **116**, 567-576 (1994).
[8] A. Ikonomopoulos and A. Endou, Wavelet decomposition and radial basis function networks for system monitoring, IEEE Transactions on Nuclear Science **45**, 2293-2301 (1998).
[9] D. W. C. Ho, P. A. Zhang, J. Xu, Fuzzy wavelet networks for function learning, IEEE Transactions on Fuzzy Systems **9**, 200-211 (2001).
[10] Q. Zhang, Using wavelet networks in nonparametric estimation, IEEE Transactions on Neural Networks **8**, 227-236 (1998).
[11] F. J. Lin, R. J. Wai, M. P. Chen, Wavelet neural network control for linear ultrasonic motor drive via adaptive sliding-mode technique, IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control **50**, 686-698 (2003).
[12] Y. C. Huang and C. M. Huang, Evolving wavelet networks for power transformer condition monitoring, IEEE Transactions on Power Delivery **17**, 412-416 (2002).
[13] L. Jiao, J. Pan, Y. Fang, Multiwavelet neural network and its approximation properties, IEEE Transactions on Neural Networks **12**, 1060-1066 (2001).
[14] S. Santini, A. D. Bimbo, R. Jain, Block-structured recurrent neural networks, Neural Networks **8**, 135-147 (1995).
[15] C. H. Lee and C. C. Teng, Identification and control of dynamic systems using recurrent fuzzy neural networks, IEEE Transactions on Fuzzy Systems **8**, 349-366 (2000).
[16] F. J. Lin, C. H. Lin, P. H. Shen, Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive, IEEE Transactions on Fuzzy Systems **9**, 751-759 (2001).
[17] J. H. Kim, D. T. College, A. Gun, G. Do, Fuzzy model based predictive control, Proc. IEEE Int. Conf. Fuzzy Systems **1**, 405-409 (1998).
[18] J. L. Elman, Finding structure in time, Cognitive Science **14**, 179-211 (1990).
[19] C. F. Juang and C. T. Lin, A recurrent self-organizing neural fuzzy inference network, IEEE Transactions on Neural Networks **10**, 828-845 (1999).
[20] C. F. Juang, A TSK-Type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms, IEEE Transactions on Fuzzy Systems **10**, 155-170 (2002).
[21] G. Chen, Y. Chen, H. Ogmen, Identifying chaotic system via a wiener-type cascade model, IEEE Transactions on Control Systems **17**, 29-36 (1997).

Cheng-Jian Lin received the Ph.D. degrees in electrical and control engineering from the National Chiao-Tung University, Taiwan, R.O.C., in 1996. Currently, he is a Distinguished Professor of Computer Science and Information Engineering Department, National Chin-Yi University of Technology, Taichung County, Taiwan, R.O.C. His current research interests are soft computing, pattern recognition, intelligent control, image processing, bioinformatics, and Android/iPhone program design. Dr. Lin is an Editorial Board of many Journal. Dr. Lin has received several honors and awards.

Chun-Cheng Peng granted his Ph.D. degree in computer science from the University of London, UK, in 2011. His main research interests include nonmonotone learning, database systems, applications in artificial intelligence and big-data processing.

Cheng-Hung Chen received the Ph.D. degree in electrical and control engineering from the National Chiao-Tung University, Taiwan, in 2008. Currently, he is an Associate Professor of Electrical Engineering Department, National Formosa University, Yunlin County, Taiwan. His current research interests are fuzzy systems, neural networks, evolutionary algorithms, intelligent control, and evolutionary robots. He has authored or coauthored more than 50 papers published in the referred journals and conference proceedings.

Hsueh-Yi Lin was born in May 10, 1959 in I-Lan County, Taiwan. He obtained Master (1991) degree in Computer Science from the Northrop University, California USA. Currently, he is an Assistant Professor of Department of Computer Science and Information Engineering of National Chin-Yi University of Technology. His research interests in image processing and computer network.