Identification and Prediction of Dynamic Systems Using an Interactively Recurrent Self-Evolving Fuzzy Neural Network

Yang-Yin Lin, Jyh-Yeong Chang, *Member, IEEE*, and Chin-Teng Lin, *Fellow, IEEE*

***Abstract*— This paper presents a novel recurrent fuzzy neural network, called an interactively recurrent self-evolving fuzzy neural network (IRSFNN), for prediction and identification of dynamic systems. The recurrent structure in an IRSFNN is formed as an external loops and internal feedback by feeding the rule firing strength of each rule to others rules and itself. The consequent part in the IRSFNN is composed of a Takagi–Sugeno–Kang (TSK) or functional-link-based type. The proposed IRSFNN employs a functional link neural network (FLNN) to the consequent part of fuzzy rules for promoting the mapping ability. Unlike a TSK-type fuzzy neural network, the FLNN in the consequent part is a nonlinear function of input variables. An IRSFNNs learning starts with an empty rule base and all of the rules are generated and learned online through a simultaneous structure and parameter learning. An on-line clustering algorithm is effective in generating fuzzy rules. The consequent update parameters are derived by a variable-dimensional Kalman filter algorithm. The premise and recurrent parameters are learned through a gradient descent algorithm. We test the IRSFNN for the prediction and identification of dynamic plants and compare it to other well-known recurrent FNNs. The proposed model obtains enhanced performance results.**

***Index Terms*— Dynamic sequence prediction, fuzzy identification, on-line fuzzy clustering, recurrent fuzzy neural networks.**

I. Introduction

DYNAMIC systems depend on past inputs, past outputs, or both, and identification and modeling of such systems are not as straightforward as that for static/algebraic systems. For dynamic system processing, practical problems are encountered in a variety of areas, such as control, pattern recognition, time series prediction, and signal processing. Recently, the combination of recurrent structures and fuzzy neural networks has become popular in identifying and recognizing temporal behaviors [1]–[16]. Therefore, recurrent structures enable effectively address temporal sequences responding to memory information from prior system states.

Manuscript received April 10, 2012; accepted November 27, 2012. Date of publication December 31, 2012; date of current version January 11, 2012. This work was supported in part by the Aiming for the Top University Plan of National Chiao Tung University, the Ministry of Education, Taiwan, under Contract 101W963, and the UST-UCSD International Center of Excellence in Advanced Bio-engineering sponsored by the Taiwan National Science Council I-RiCE Program under Grant NSC-100-2911-I-009-101.

The authors are with the Institute of Electrical Control Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: oliver.yylin@gmail.com; jychang@mail.nctu.edu.tw; ctlin@mail.nctu.edu.tw).

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

Digital Object Identifier 10.1109/TNNLS.2012.2231436

In contrast with pure feed-forward fuzzy neural network (FNN), we have to know the number of lagged inputs and outputs in advance, and feed these lagged values as feed-forward FNN input. The exact order of a dynamic system is usually unknown, and thus, we do not know the number of lagged values to provide. Moreover, the lagged values increase input dimensions and result in a larger network size. Apparently, the use of a feed-forward FNN is unsuitable for constructing dynamic system. Therefore, some recurrent fuzzy neural networks (RFNNs) have already been proposed [1]–[16] for solving the temporal characteristics of dynamic systems, and have been shown to outperform feed-forward FNNs and recurrent neural networks.

One important purpose is to design a consequent part of FNN, which is able to impact performance on using different types. Researchers usually use two types of fuzzy if-then rules and fuzzy reasoning employed, i.e., Mamdani-type and Takagi–Sugeno–Kang (TSK)-type. For Mamdani-type fuzzy neural networks [2], [6], [17]–[19], the minimum fuzzy implication is adopted in fuzzy reasoning. For TSK-type fuzzy neural networks [5], [9], [14], [20], [21], the consequent part of each rule is a linear function of input variables. Several studies [14], [20], [21] indicate that the performance of a feedforward TSK-type fuzzy network in network size and learning accuracy is superior to those of Mamdani-type fuzzy networks. A feedforward TSK-type fuzzy network appears to have more free parameters to adjust input space mapping. However, each consequent part of each fuzzy rule in a standard TSK-type fuzzy neural network does not take full advantage of the mapping capabilities of local approximation by rule hyper-planes. Therefore, several studies [22]–[28] consider trigonometric functions to replace the traditional TSK-type fuzzy reasoning and also obtain the better performance.

In this view, the functional-link neural networks (FLANN) [22], [23] have been proposed using trigonometric functions to construct consequent part. The functional expansion increases the dimensionality of the input vector and thus, creation of nonlinear decision boundaries in the multidimensional space and identification of complex nonlinear functions become simple with this network. It seems to be more efficient, based on these results, to include the functional-link fuzzy rules into the design of recurrent fuzzy network.

With above mentioned motivations, this paper presents the combination of a novel recurrent structure and a FLANN to construct the consequent part, called an interactively recurrent

self-evolving fuzzy neural network (IRSFNN), for dynamic system identification and prediction. The proposed IRSFNN contains four major contributions as follows.

1) A novel recurrent structure with interaction feedback incorporates the advantages of local feedback and global feedback. The global feedback in the proposed network means that the necessary information is obtained from the other fuzzy rules. Local source (a rule gets feedback from itself only) is not sufficient to represent the necessary information. For a more efficient flow, external (global) feedback for reimbursing the local information is derived.
2) Many studies [1]–[13] have only considered the past states in recurrent structure, which is insufficient without referring to current states. Previous studies [14], [29] provided strong evidence that compatibly use past and current states to be more desirable. Therefore, the proposed model depends on current states along with previous states.
3) We use the FLNN to replace the traditional TSK-type fuzzy reasoning, and compare their performance. As explained before, the functional expansion increases the dimensionality of the input pattern and thus, creation of nonlinear decision boundaries in the multidimensional space and identification of complex nonlinear functions become simple with this network.
4) We use innovative learning algorithms for the structure and parameters of the system. For structure learning, all of the rules and fuzzy sets are generated on-line in an IRSFNN, which helps to automate rule generation. We do not need to set any initial IRSFNN structure in advance. The antecedent part and recurrent parameters are learned by gradient descent algorithm. The consequent parameters in an IRSFNN are tuned using a variable-dimensional Kalman filter algorithm. This algorithm handles inputs with variable dimensions, a phenomenon caused by incremental rules during the structure learning process.

We then conduct several simulations to assess the IRSFNN performance, and compare the IRSFNN with other existing models.

The rest of this paper is organized as follows. Section II illustrates brief survey of some existing methods; Section III introduces the FLANN and IRSFNN structure. Section IV introduces an interpretation of structure and parameter learning methods for the IRSFNN. Section V presents results from two types of dynamic system identification and prediction of time series problems, including the Henon chaotic sequence, Mackey–Glass time series, and Box–Jenkins time series; and finally, Section VI offers a conclusion.

II. Brief Survey of Some Existing Methods

Recently, considerable research has been devoted toward these developing RFNNs, and these networks can be separated into two major categories. One category of recurrent FNNs in [1]–[9] and [16], uses global feedbacks. In [2], a recurrent self-organizing neural fuzzy inference network (RSONFIN) computes the values of the internal feedback variables using all rule firing strengths and the consequent parts are fuzzy sets. The recurrent structure in the RSONFIN just considers past state. For parameter learning, the RSONFIN uses gradient descent algorithm to tune free parameters. The authors in [3] and [4] proposed an output-RFNN where the output values are fed back as input values. In [7], the TSK-type recurrent fuzzy network's (TRFNs) structure is similar to an RSONFIN. The recurrent neuron-fuzzy network in [9] feeds back the network output values not only globally to all the rule inputs, but also locally to the consequent part of each rule, in the form of the autoregressive moving average with exogenous inputs model. The recurrent high-order neural network (RHONN) [16] trained with an extended Kalman filter algorithm was proposed for optimal control of nonlinear systems.

The other approach of recurrent FNNs [10]–[14] uses feedback loops from internal state variables as its recurrence structure. The design of local recurrent structures seems to be simpler than that of global recurrent structures, and also obtains superior performance. In [10] and [11], the recurrent property is achieved by feeding the output of each membership function (MF) back to itself; thus each membership value is only influenced by its previous value. The recurrent property in study [14], a recurrent self-evolving fuzzy neural network with local feedback (RSEFNN-LF) is achieved by locally feeding the output of temporal firing strength back to itself; thus, temporal firing strength is influenced by current and past states.

As mentioned earlier, many researchers frequently use Mandani-type or TSK-type to construct consequent part of fuzzy rules. Many studies indicate that TSK-type fuzzy systems significantly outperform Mandani-type fuzzy systems. However, TSK-type fuzzy neural network does not take full advantage of the mapping capabilities of local approximation by rule hyper-planes. In order to overcome this problem, our proposed model employs the FLANN [22], [23] to strength the mapping ability of input space. Therefore, nonlinear function (trigonometric function) to the consequent part shall be able to effectively discriminate in mapping input space. Previous studies [22]–[28] indicated that the use of trigonometric function obtains better performances than the use of TSK-type. As a result, in this paper, the marriage of a novel recurrent structure and functional-link-based NN is a significant research for addressing the temporal problems.

III. IRSFNN Structure

This section introduces the structure of the FLNN and multiple-input-single-output IRSFNN. The recurrent structure in the IRSFNN uses interaction feedback that has the ability to capture critical information from other rules. The consequent part of each recurrent fuzzy rule is functional link and executes a nonlinear model. Next, we have described the structure of FLNN.

A. Functional Link Artificial Neural Network

The functional link artificial neural network (FLANN) is basically a single layer structure in which nonlinearity is

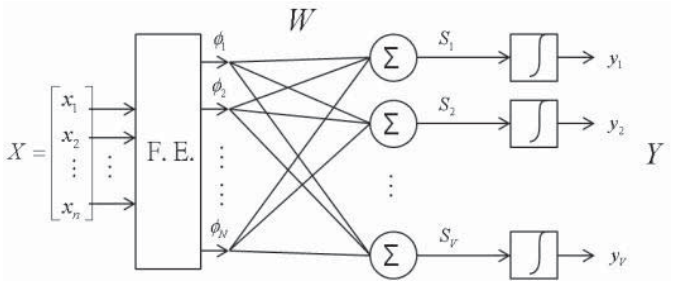

Fig. 1. Structure of FLANN.

introduced by enhancing the input pattern with nonlinear functional expansion. Therefore, the FLANN structure considers trigonometric functions. Fig. 1 shows the structure of FLANN, where each of the input patterns is passed through a functional expansion block yielding a corresponding N-dimensional expanded vector. Suppose that for the input pattern X is of a 2-D input $(x_1,\ x_2)$, the expanded inputs are using trigonometric functions to be taken. The expanded input variables can be denoted as $\vec{\varphi} = (1,\ x_1,\ \sin(\pi x_1),\ \cos(\pi x_1),\ x_2,\ \sin(\pi x_2),$ and $\cos(\pi x_2))$.

The theory of the FLANN for multidimensional function approximation has been discussed and analyzed below [22], [23].

Let us consider a set of basis functions $B = \{\varphi_k \in \mathbf{\Phi}(A)\}_{k\in \mathrm{K}}$, with the following properties: 1) $\varphi_1 = 1$; 2) the subset $B_j = \{\varphi_k \in B\}_{k=1}^{N}$ is a linearly independent set, that is, if $\sum_{k=1}^{N} \varphi_k w_k = 0$, then $w_k = 0$ for $k = 1, \ldots, j$; and 3) $\sup_j \left[\sum_{k=1}^{j} \|\varphi_k\|_A^2\right]^{1/2} < \infty$. Next, $B_N = \{\varphi_k\}_{k=1}^{N}$ is a set of a set of basis functions to be considered, as shown in Fig. 1. Hence, output of functional expansion block is composed by $(\varphi_1,\ \varphi_2,\ \ldots,\ \varphi_N) \in B_N$ with the following input–output relationship for the j-th output:

$$\hat{y}_j = \rho(S_j); \quad S_j = \sum_{k=1}^{N} \phi_{kj} w_k(X) \tag{1}$$

where $X \in A \subset \Re^n$, that is, $X = (x_1, x_2, \ldots, x_n)^T$ is the input dimension and $W = (w_{j1}, w_{j2}, \ldots, w_{jN})^T$ is the weight vector associated with the j-th output of the FLANN. The vector S is a matrix of linear outputs of the FLANN, and the output vector is $\hat{y} \in \Re^V$, that is, $\hat{y} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_V)^T$. The nonlinear function can be denoted as

$$\rho\,(\cdot) = \tanh\,(\cdot). \tag{2}$$

In the IRSFNN model, the corresponding weights of functional link bases do not exist in the initial state, and the amount of the corresponding weights of functional link bases generated by the online learning algorithm is consistent with the number of fuzzy rules. Section III describes the self-evolving technology.

B. IRSFNN Structure

This section describes the IRSFNN model that employs FLANN to the consequent part of the IRSFNN for enhancing network's performance. Fig. 2 shows the proposed six-layered IRSFNN structure. The detailed function of each layer is discussed next.

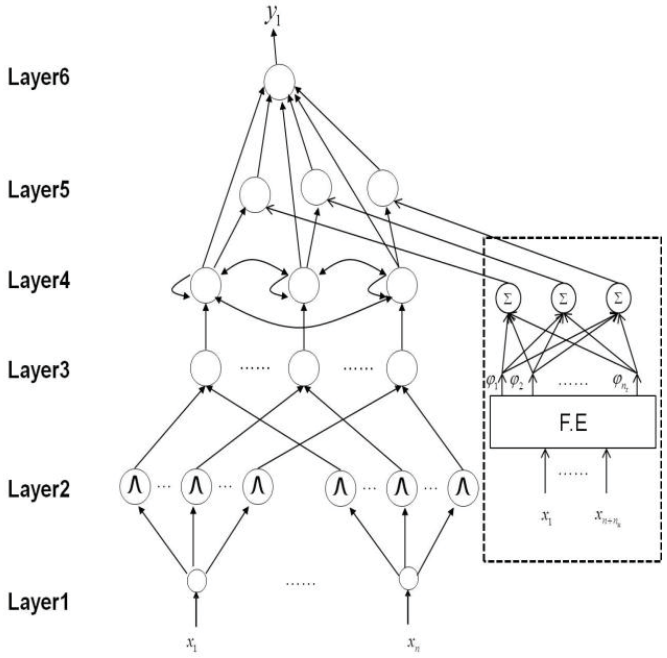

Fig. 2. Structure of the proposed IRSFNN model, where each recurrent fuzzy rule in layer 4 forms a locally and globally recurrent structure and each node in layer 5 combines functional-link-based framework.

For a clear understanding of the mathematical function of each node, we will describe function relationship between each layer. The net input to the i-th node in layer l is represented as $u_i^{(l)}$ and the output value is represented as $O_i^{(l)}$.

Layer 1 (Input Layer): The inputs are crisp values and $\overrightarrow{x} = (x_1, \ldots, x_n)$ are fed as inputs to this layer. This is in contrast to feed-forward FNNs where both current and past states are fed as inputs to input layer when such networks are used to model time-varying systems. Weight requiring adjustment in this layer is absent.

Layer 2 (Fuzzification Layer): Each node in this layer defines a Gaussian MF and performs a fuzzification operation. For the i-th fuzzy set A_j^i on the input variable x_j, $j = 1, \ldots, n$, a Gaussian MF is computed by (3)

$$\mu_j^i(x_j) = O_i^{(2)} = \exp\left\{-\frac{1}{2}\left(\frac{u_j^{(2)} - m_j^i}{\sigma_j^i}\right)^2\right\}, \text{ and}$$
$$u_j^{(2)} = O_j^{(1)}. \tag{3}$$

Layer 3 (Spatial Firing Layer): Each node in this layer represents one fuzzy rule that computes the firing strength. Because this layer does not depend on any temporal input, we call this layer "spatial" to distinguish it from the "temporal" firing strength computed in the next layer. For the obtained spatial firing strength ϕ^i, each node performs a fuzzy meet operation on inputs it receives from layer 2 using an algebraic product operation.

There are M nodes in this layer, and the spatial firing strength is computed as

$$\varphi^i = O_i^{(3)} = \prod_{j=1}^{n} u_j^{(3)}, \quad \text{and} \quad u_j^{(3)} = O_j^{(2)}. \tag{4}$$

As in (2), M is the total number of rules.

Layer 4 (Temporal Firing Layer): Each node in this layer is a recurrent rule node, which formulates an internal feedback (self-loop) and external interaction feedback loop. The output of a recurrent rule node is a temporal firing strength that depends not only on current spatial firing strength but also on the previous temporal firing strength. The temporal firing strength is a linear combination function expressed as

$$O_i^{(4)} = \sum_{k=1} (\lambda_{ik}^q \cdot O_k^{(4)}(t-1)) + (1-\gamma_i^q) \cdot u_i^{(4)}, \quad \text{and}$$
$$u_i^{(4)} = O_i^{(3)} \tag{5}$$

that is

$$\psi_i^q(t) = \sum_{k=1} (\lambda_{ik}^q \cdot \psi_k^q(t-1)) + (1-\gamma_i^q) \cdot \varphi^i(t),$$
$$i = 1, \ldots, M \ \text{ and } \ q = 1, \ldots, n_O \tag{6}$$

where $\gamma_i^q = \sum_{k=1}^{M} \lambda_{ik}^q$ and $\lambda_{ik}^q = (C_{ik}^q)/M$ $(0 \le C_{ik}^q \le 1)$ is the rule interaction weight between itself and other rules. For the updated recurrent weights, the proposed approach uses a gradient descent algorithm to derive the optimal values. The recurrent weights λ_{ik}^q determine the compromised ratio between the current and previous inputs to the network outputs.

Layer 5 (Consequent Layer): Each node in this layer is an optional node, called a consequent layer, and can be TSK-type or functional-link-based fuzzy rules. The weight of the link from a node in layer 4 to one in layer 5 is a_{i0}^q, for $q = 1, \ldots, n_o$ and $i = 1, \ldots, M$. For the TSK-type IRSFNN, the node output is a linear combination of current input states $x_1, \ldots, x_n$. The output of TSK-type $\overline{v}_i^q$ of the i-th rule node connecting to the q-th output variable is computed as follows:

$$\overline{v}_i^q = \bar{O}_i^{(5)} = \sum_{j=0}^{n} a_{ij}^q \cdot u_j^{(4)}, \ \text{ and } \ u_j^{(4)} = O_j^{(1)} \tag{7}$$

where $x_0 \equiv 1$.

For the functional-link-based IRSFNN, the output uses a functional expansion as given by the trigonometric polynomial basis function $[x_1 \sin(\pi x_1) \cos(\pi x_1) x_2 \sin(\pi x_2) \cos(\pi x_2)]$ for 2-D input variables. The output of functional-link-based $\tilde{v}_i^q(t)$ is expressed by

$$\tilde{v}_i^q = \tilde{O}_i^{(5)} = \sum_{k=0}^{n_t} a_{ik}^q \cdot \phi_k, \qquad n_t = 3 \times (n + n_u) \tag{8}$$

where $\phi_0 \equiv 1$.

The coefficient n_u denotes lag numbers of system output or control input. If we do not use extra lagged values ($n_u = 0$), $\varphi_k = (x_1, \sin(\pi x_1), \cos(\pi x_1), \ldots, x_n, \sin(\pi x_n), \cos(\pi x_n))$ and $k = 1, \ldots, n_t$. The constant n_t is an amount of basis expansion according to input variables.

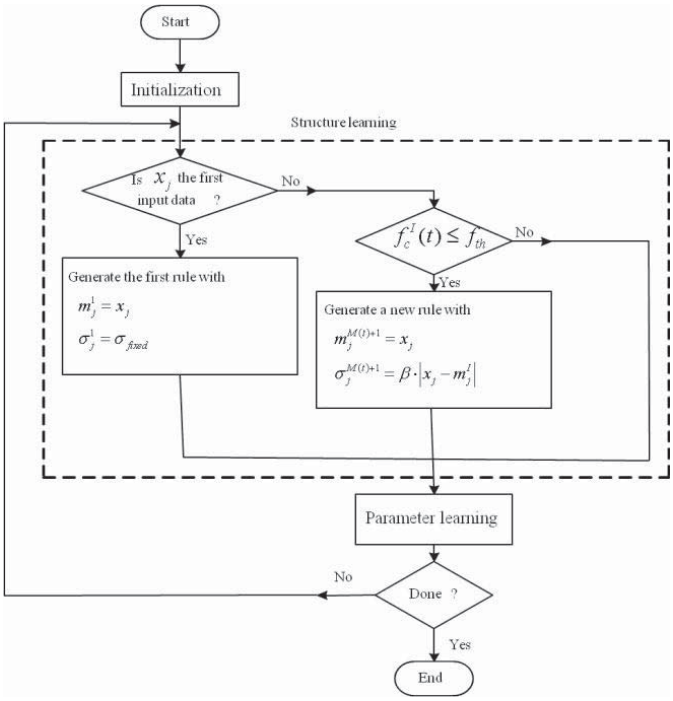

Fig. 3. Flowchart of the structure and parameter learning of the IRSFNN.

Layer 6 (Output Layer): Each node in this layer corresponds to one output variable. For defuzzification operations, the q-th output layer node computes the network output variable y_q by using the weighted average method.

For the TSK-type IRSFNN, the output can be expressed as

$$y_q = O^{(6)} = \frac{\sum_{i=1}^{M} O_i^{(4)} \overline{O}_i^{(5)}}{\sum_{i=1}^{M} O_i^{(4)}} = \frac{\sum_{i=1}^{M} \psi_i^q(t) \cdot \sum_{j=0}^{n} a_{ij}^q \cdot x_j}{\sum_{i=1}^{M} \psi_i^q(t)},$$
$$q = 1, \ldots, n_o \tag{9}$$

where $\overline{v}$ denotes the consequent value and a denotes the parameters.

For the functional-link-based IRSFNN, the output is

$$y_q = O^{(6)} = \frac{\sum_{i=1}^{M} O_i^{(4)} \tilde{O}_i^{(5)}}{\sum_{i=1}^{M} O_i^{(4)}} = \frac{\sum_{i=1}^{M} \psi_i^q(t) \cdot \sum_{k=0}^{n_t} a_{ik}^q \cdot \phi_j}{\sum_{i=1}^{M} \psi_i^q(t)},$$
$$q = 1, \ldots, n_o \tag{10}$$

where $\tilde{v}$ denotes the consequent value and a denotes the parameters.

IV. IRSFNN Learning

In this section, two-phase learning is used for constructing the IRSFNN. Initially, no rules are in an IRSFNN. All of the recurrent fuzzy rules evolve from the simultaneous structure and parameter learning after receiving each piece of training data. Fig. 3 presents flowchart of the IRSFNNs learning scheme. The proposed model uses efficient structure learning

to evolve fuzzy rules and fuzzy sets on-line. The parameter learning phase describes the use of a gradient descent algorithm and a variable-dimensional Kalman filter algorithm. The following sections introduce the structure and parameter learning algorithms explicitly.

A. Structure Learning

The first task in structure learning is to determine whether a new rule should be extracted from the training data and to determine the number of fuzzy sets in the universe of discourse of each input variable because one cluster in the input space corresponds to one potential fuzzy rule, in which m_j^i represents the mean and σ_j^i represents the variance of that cluster. The spatial firing strength φ^i in (4) is used to determine whether a new rule should be generated. The first incoming data point $\mathbf{x}$ is used to generate the first fuzzy rule, and the mean and width of the fuzzy MFs associated with this rule are set as:

$$m_j^1 = x_j \text{ and } \sigma_j^1 = \sigma_{\text{fixed}}, \qquad j = 1, \ldots, n \tag{11}$$

where σ_{fixed} is a predefined value (we use $\sigma_{\text{fixed}} = 0.3$ in this paper) that determines the width of the memberships associated with a new rule. For subsequent new incoming data $\mathbf{x}(t)$, we find

$$I = arg \max_{1<i<M(t)} f_{c^i}(t) \tag{12}$$

where M(t) is the number of existing rules at time t. If $f_c^I(t) \leq f_{\text{th}}$($f_{\text{th}}$ is a pre-specified threshold), then a new fuzzy rule is generated and $M(t+1) = M(t) + 1$. In this approach, if the present data do not match well according to the existing rules, then a new rule is evolved. We also use the same procedure to assign the mean of fuzzy sets as we have done for first rule. For a new rule, the mean and width of corresponding fuzzy sets are defined as

$$m_j^{M(t)+1} = x_j \text{ and } \sigma_j^{M(t)+1} = \beta \cdot \left| x_j - m_j^I \right|, \quad j = 1, \ldots, n \tag{13}$$

where β is an overlap coefficient. Equation (13) indicates that the initial width is equal to the Euclidean distance between current input data x and the center of the best matching rule for this data point times an overlapping parameter β. In this paper, β is set to 0.5, so that the width of new fuzzy set is half of the Euclidean distance from the best matching center, and a suitable overlap between adjacent rules is realized.

B. Parameter Learning

In addition with the structure, all free parameters in an IRSFNN are also learned, including those newly generated and previously existing. For clarification, we consider the single-output case and define the objective to minimize the error function as

$$E = \frac{1}{2}\left[y_q(t) - y_d(t)\right]^2 \tag{14}$$

where $y_q(t)$ represents the IRSFNN output and $y_d(t)$ represents the desired output. Parameters in the consequent part of the TSK-type IRSFNN are learned based on the variable-dimensional Kalman filter algorithm as discussed in [14]. According to [14], (10) of a functional-link-based IRSFNN can be re-written as

$$y_q = \vec{\psi}(t)_{\text{FuL}}^T \vec{a}_{\text{FuL}} \tag{15}$$

where

$$\vec{\psi}(t)_{\text{FuL}}^T = \left[\overbrace{\frac{\psi_1^q(t)}{\sum_{i=1}^M \psi_i^q(t)}\varphi_0, \ldots, \frac{\psi_1^q(t)}{\sum_{i=1}^M \psi_i^q(t)}\varphi_{n_t}}^{n_t+1}, \ldots, \overbrace{\frac{\psi_M^q(t)}{\sum_{i=1}^M \psi_i^q(t)}\varphi_0, \ldots, \frac{\psi_M^q(t)}{\sum_{i=1}^M \psi_i^q(t)}\varphi_{n_t}}^{n_t+1} \right] \in \Re^{1\times(n_t+1)\times M} \tag{16}$$

and

$$\vec{a}_{\text{FuL}} = \left[a_{10}^q, \ldots, a_{1n_t}^q, \ldots, a_{M0}^q, \ldots, a_{Mn_t}^q\right]^T \in \Re^{M\times(n_t+1)\times 1} \tag{17}$$

and

$$[\varphi_0, \varphi_1, \ldots, \varphi_{n_t}] = [1, x_1, \sin(\pi x_1), \cos(\pi x_1), \ldots, x_n, \sin(\pi x_n), \cos(\pi x_n)]. \tag{18}$$

The consequent parameter vector $\vec{\mathbf{a}}_{\text{FuL}}$ is updated by executing the following variable-dimensional Kalman filtering algorithm:

$$\begin{aligned} \vec{a}_{\text{FuL}}(t+1) &= \vec{a}_{\text{FuL}}(t) + S(t+1)\vec{\psi}_{\text{FuL}}(t+1)(y_d(t+1) \\ &\quad - \vec{\psi}_{\text{FuL}}(t+1)\vec{a}_{\text{FuL}}(t)) \\ S(t+1) &= \frac{1}{\kappa}\left[S(t) - \frac{S(t)\vec{\psi}_{\text{FuL}}(t+1)\vec{\psi}_{\text{FuL}}^T(t+1)S(t)}{\kappa + \vec{\psi}_{\text{FuL}}^T(t+1)S(t)\vec{\psi}_{\text{FuL}}(t+1)}\right] \end{aligned} \tag{19}$$

where κ is a forgetting factor and lies in [0, 1] ($\kappa = 0.99995$ in this paper). Once a new rule is generated, the dimension of the vectors $\vec{a}_{\text{FuL}}$, $\vec{\psi}_{\text{FuL}}$, and the matrix S increases accordingly. When a new rule evolves at time $t+1$, the new vector $\vec{\psi}_{\text{FuL}}(t+1)$ becomes

$$\vec{\psi}(t+1)_{\text{FuL}}^T = \left[\overbrace{\frac{\psi_1^q(t)}{\sum_{i=1}^M \psi_i^q(t)}\varphi_0, \ldots, \frac{\psi_1^q(t)}{\sum_{i=1}^M \psi_i^q(t)}\varphi_{n_t}}^{n_t+1}, \ldots, \overbrace{\frac{\psi_{M+1}^q(t)}{\sum_{i=1}^M \psi_i^q(t)}\varphi_0, \ldots, \frac{\psi_{M+1}^q(t)}{\sum_{i=1}^M \psi_i^q(t)}\varphi_{n_t}}^{n_t+1} \right] \in \Re^{1\times(n_t+1)\times(M+1)}. \tag{20}$$

An IRSFNN augments $\vec{a}_{\text{FuL}}(t)$ and $S(t)$ on the right-hand side of (16) as follows:

$$\vec{a}_{\text{FuL\_new}} = \left[\vec{a}_{\text{FuL}}, a_{(M+1)0}^q, \ldots, a_{(M+1)n_t}^q\right]^T \in \Re^{(M+1)\times(n_t+1)\times 1} \tag{21}$$

and

$$S_{\text{new}}(t) = \text{blockdiag}[S(t)C \cdot I] \in \Re^{(M+1)(n_t+1)\times(M+1)(n_t+1)} \tag{22}$$

where C is a large positive constant (we use $C = 10$).

TABLE I
PERFORMANCE OF IRSFNN AND OTHER RECURRENT MODELS FOR DYNAMIC SYSTEM IDENTIFICATION IN EXAMPLE 1

Models	RSONFIN [2]	WRFNN [11]	HO-RNFS [6]	TRFN [7]	RSEFNN-LF [14]	IRSFNN (TSK)	IRSFNN (FuL)
Rules	4	5	3	3	4	3	3
Number of Parameters	36	55	45	33	32	30	42
Training RMSE	0.025	0.064	0.054	0.032	0.020	0.015	0.011
Test RMSE	0.078	0.098	0.082	0.047	0.040	0.036	0.031

∗FuL denotes Functional−Link−based.

TABLE II
PERFORMANCE OF IRSFNN AND OTHER RECURRENT MODELS FOR DYNAMIC SYSTEM IDENTIFICATION IN EXAMPLE 2

Models	RSONFN [2]	WRFNN [11]	TRFN [7]	RSEFNN-LF [14]	IRSFNN (TSK)	IRSFNN (FuL)
Rules	6	5	3	4	3	2
Number of Parameters	36	55	33	30	30	26
Training RMSE	0.03	0.057	0.007	0.016	0.014	0.011
Test RMSE	0.06	0.083	0.031	0.028	0.026	0.022

This paper uses resetting operations to keep S bounded and to avoid divergence problems. After a period of training, the matrix S is re-set as $C \cdot I$. Simulation results in Section IV show that the learning of the IRSFNN achieves good training and test performance. A gradient descent algorithm tunes the antecedent parameters of the IRSFNN. This gradient descent algorithm is performed once for each piece of incoming datum.

By using a gradient descent algorithm for the updated recurrent weights, we have

$$\lambda_{ik}^{q}(t+1) = \lambda_{ik}^{q}(t) - \eta \frac{\partial E}{\partial \lambda_{ik}^{q}} \tag{23}$$

where η is the learning rate and

$$\frac{\partial E}{\partial \lambda_{ik}^{q}} = \frac{\partial E}{\partial y_q}\frac{\partial y_q}{\partial \psi_i^q}\frac{\partial \psi_i^q}{\partial \lambda_{ik}^{q}} = \frac{(y_q - y_d)\cdot(v_i^q - y_q)\cdot(\psi_k^q(t-1) - \phi^i(t))}{\sum_{i=1}^{M}\psi_i^q(t)}. \tag{24}$$

The antecedent part of parameter m_j^i is updated as

$$m_j^i(t+1) = m_j^i(t) - \eta \frac{\partial E}{\partial m_j^i} \tag{25}$$

where

$$\frac{\partial E}{\partial m_j^i} = \frac{\partial E}{\partial y_q}\frac{\partial y_q}{\partial \psi_i^q}\frac{\partial \psi_i^q}{\partial \phi^i}\frac{\partial \phi^i}{\partial \mu_j^i}\frac{\partial \mu_j^i}{\partial m_j^i} = (y_q - y_d)\cdot\frac{(v_i^q - y_q)}{\sum_{i=1}^{M}\psi_i^q(t)} \cdot (1-\gamma_i^q)\cdot\phi^i\cdot\frac{2(x_j - m_j^i)}{(\sigma_j^i)^2}. \tag{26}$$

The antecedent part of parameter σ_j^i is updated as

$$\sigma_j^i(t+1) = \sigma_j^i(t) - \eta \frac{\partial E}{\partial \sigma_j^i} \tag{27}$$

where

$$\frac{\partial E}{\partial \sigma_j^i} = \frac{\partial E}{\partial y_q}\frac{\partial y_q}{\partial \psi_i^q}\frac{\partial \psi_i^q}{\partial \phi^i}\frac{\partial \phi^i}{\partial \mu_j^i}\frac{\partial \mu_j^i}{\partial \sigma_j^i} = (y_q - y_d)\cdot\frac{(v_i^q - y_q)}{\sum_{i=1}^{M}\psi_i^q(t)} \cdot (1-\gamma_i^q)\cdot\phi^i\cdot\frac{2(x_j - m_j^i)^2}{(\sigma_j^i)^3}. \tag{28}$$

V. SIMULATION

This section presents five examples to assess the performance of the proposed IRSFNN. These simulation studies include two types of dynamic system problems (Examples 1, 2) and three types of prediction problems (Examples 3–5). These examples are also used to compare the performance of the IRSFNN with those of existing recurrent FNNs. For dynamic system identification, the recurrent structure in the proposed approach shows the advantages, as listed in Tables I–V.

A. Example 1 (Dynamic System Identification)

This example uses an IRSFNN to identify a nonlinear dynamic system, which is a nonlinear plant with multiple time delays that has been studied in [7]. The dynamic system is described by the following difference equation:

$$y_p(t+1) = f(y_p(t), y_p(t-1), y_p(t-2), u(t), u(t-1)) \tag{29}$$

TABLE III
PERFORMANCE OF IRSFNN AND OTHER RECURRENT MODELS FOR CHAOTIC SEQUENCE PREDICTION IN EXAMPLE 3

Models	RFNN [10]	WRFNN [11]	TRFN-S [7]	RSEFNN-LF [14]	IRSFNN (TSK)	IRSFNN (FuL)
Rules	15	7	6	9	4	3
Number of Parameters	60	70	66	45	32	40
Training RMSE	0.463	0.191	0.028	0.032	0.017	0.016
Test RMSE	0.469	0.188	0.027	0.023	0.015	0.014

TABLE IV
PERFORMANCE OF IRSFNN AND OTHER MODELS FOR MACKEY–GLASS CHAOTIC SEQUENCE PREDICTION PROBLEM IN EXAMPLE 4

Models	Rules	Number of Parameters	Train RMSE	Test RMSE
D-FNN [12]	10	100	–	0.0082
G-FNN [1]	10	90	–	0.0056
Recurrent ANFIS [31]	–	–	–	0.0013
SEELA [32]	9	198	0.0067	0.0068
SuPFuNIS [33]	10	94	–	0.0057
TRFN-S [7]	5	95	–	0.0124
CSPSO [34]	10	104	–	0.0064
FLNFN-CCPSO [35]	–	–	0.0083	0.0084
LLWNN+Hybrid [36]	–	110	0.0033	0.0036
FWNN [37]	16	128	0.0023	0.0023
RSEFNN-LF [14]	9	94	0.0032	0.0031
IRSFNN (TSK)	5	90	0.0040	0.0039
IRSFNN (FuL)	4	100	0.0002	0.0002

TABLE V
PERFORMANCE OF IRSFNN AND OTHER MODELS FOR BOX–JENKINS PREDICTION IN EXAMPLE 5

Models	Rules	Number of Parameters	Train RMSE	Test RMSE
HyFIS [38]	–	–	–	0.0205
Recurrent ANFIS [31]	–	–	0.006	0.0193
TRFN-S [7]	5	65	0.0524	0.0482
TNFIS [39]	–	43	0.0245	0.0230
FuNN [40]	–	–	–	0.0226
LLWNN+Hybrid [36]	–	56	–	0.0138
FWNN [37]	9	57	0.0189	0.0279
RSEFNN-LF [14]	7	56	0.0172	0.0344
IRSFNN (TSK)	5	65	0.0121	0.0297
IRSFNN (FuL)	3	51	0.00062	0.0009

where

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) x_4}{1 + x_2^2 + x_3^2}. \tag{30}$$

The dynamic system output depends on three previous outputs and two previous inputs. In this paper, only two current values, $u(t)$ and $y_p(t)$, are fed as input to the IRSFNN input layer. Here, we do not use extra lagged values ($n_u = 0$) in the consequent part. The desired output of the IRSFNN is $y_d(t+1)$. In the training procedure of the IRSFNN, we follow

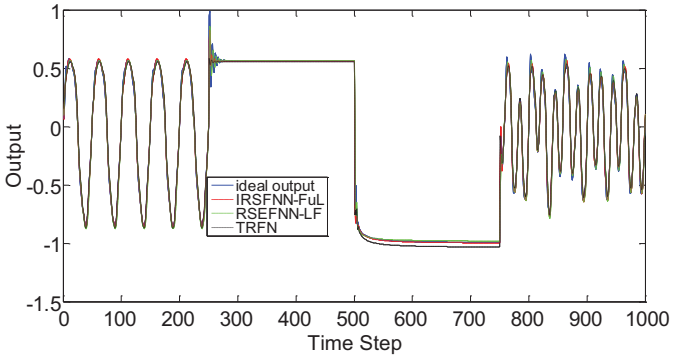

Fig. 4. Outputs of the dynamic plant (blue line), IRSFNN (red line), RSEFNN_LF (green line), and TRFN (black line) in Example 1.

the same computational protocols as in [7] and [14], i.e., we use only ten epochs, with 900 time steps in each epoch. In each epoch, the first 350 inputs are random values uniformly distributed over $[-2, 2]$ and the remaining 550 training inputs are generated from a sinusoid, $1.05\sin(\pi t/45)$.

We follow this strategy for an online training process because a similar procedure was followed in [14], where the total number of online training time steps is 9000. The structure learning threshold f_{th} decides the number of rules to be generated. After training, three rules are generated when the structure learning threshold is set to 0.01. Table I shows the root-mean-squared error (RMSE) of training data. The testing input signal $u(t)$ is guided by

$$u(t) = \begin{cases} \sin(\frac{\pi t}{25}), & t < 250 \\ 1.0, & 250 \le t < 500 \\ -1.0, & 500 \le t < 750 \\ 0.3\sin(\frac{\pi t}{25}) + 0.1\sin(\frac{\pi t}{32}) & \\ +0.6\sin(\frac{\pi t}{10}), & 750 \le t < 1000. \end{cases} \tag{31}$$

Fig. 4 shows a comparison of the actual output with the output produced by the IRSFNN for the test input. Fig. 5 shows the error difference between the actual plant output and the IRSFNN. Figs. 4 and 5 show a very good match, suggesting that IRSFNN architecture combined with the proposed system identification scheme adequately identifies the dynamic system with feedback.

Table I shows the performance of the IRSFNN compared with the other recurrent networks, including a RSONFIN [2], a wavelet RFNN (WRFNN) [11], a TSK-type TRFN [7], a HO-RNFS [6], and a RSEFNN-LF [14].

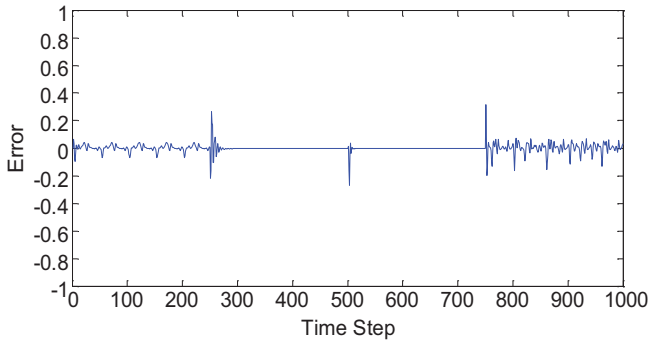

Fig. 5. Test errors between the MRIT2FNN and actual plant outputs.

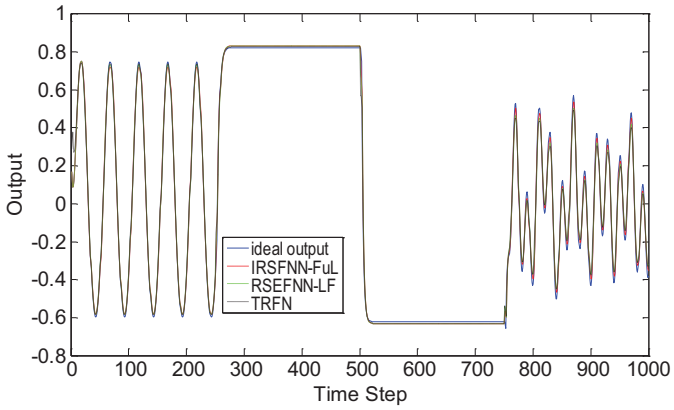

Fig. 6. Outputs of the dynamic plant (blue line) and IRSFNN-FuL (red line) in Example 2.

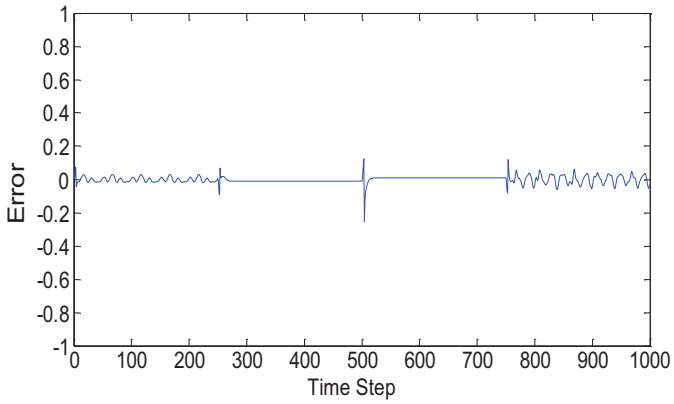

Fig. 7. Test errors between the IRSFNN-FuL and actual plant outputs.

The consequent part in the RSEFNN-LF is composed by a first-order TSK-type that performs a linear combination of input variables. As in the IRSFNN, all these networks use the same information, including the number of input variables, training data, test data, and training epochs. For a fair comparison, the total number of parameters of the IRSFNN is kept similar to that of the compared networks. The result indicates that the IRSFNN achieves better identification than the other recurrent networks.

B. Example 2 (Dynamic System Identification)

This example considers the use of the IRSFNN for dynamic system identification with longer input delays that is described by

$$y_p(t+1) = 0.72y_p(t) + 0.025y_p(t-1)u_1(t-1) + 0.01u_1^2(t-2) + 0.2u_1(t-3). \quad (32)$$

This plant is the same as the one used in [7]. This plant output depends on four previous inputs and two previous outputs. As shown in Example 1, the current variables $u(t)$ and $y_p(t)$ are fed as inputs to the IRSFNN input layer. In this example, we do not use extra lagged values ($n_u = 0$) in the consequent part. The training data and time steps are the same as those used in Example 1. When the structure learning threshold f_{th} is set to 0.05, three rules are generated. The test signal used in Example 1 is also adopted here to assess the identified system. Fig. 6 shows the outputs of the plant and the IRSFNN for these test inputs. Fig. 7 shows the test error between the outputs of the IRSFNN and the desired plant. Table II shows the number of rules, total number of parameters, and training and test RMSEs of the IRSFNN. The performance of the IRSFNN is compared with that of recurrent models, including an RSONFIN [2], a TRFN [7], a WRFNN [11], and an RSEFNN-LF [14]. These models use identical numbers of input variables, training data, test data, and training epochs as designed by the IRSFNN. For a fair comparison, the numbers of parameters in the IRSFNN have been kept similar to those in these compared models.

Apparently in Table I, the RSEFNN-LF only uses local source, which is not enough to capture critical information for the system, thus the test error of the IRSFNN_TSK is lower than that of the RSEFNN-LF, even using fewer rules. Here, we also investigate the performance comparison of the IRSFNN-TSK and the IRSFNN-FuL, and results show that the IFSFNN-FuL achieves the better performance. Finally, the results show that the test RMSEs of the IRSFNN-FuL and IRSFNN-TSK are smaller than those of the other networks.

C. Example 3 (Chaotic Series Prediction)

As introduced in [30], the IRSFNN is applied to predict the Henon chaotic sequence of a dynamic system with one delay and two sensitive parameters generated by the following equation:

$$y_p(t+1) = -P \cdot y_p^2(t) + Q \cdot y_p(t-1) + 1.0. \quad (33)$$

Equation (33), with $P = 1.4$ and $Q = 0.3$, produces a chaotic attractor. The initial states $[y_p(1), y_p(0)] = [0.4, 0.4]$ generate 2000 patterns, with the 1000 patterns used for training and the remaining 1000 patterns used for testing. In this example, we do not use extra lagged values ($n_u = 0$) in the consequent part. The training procedure uses the plant output $y_p(t+1)$ as the desired output $y_d(t+1)$. The system has a single output so that only output variable $y_p(t)$ is fed as input to the IRSFNN. The training epoch in the IRSFNN is set to 90. The structure learning threshold f_{th} is set to 0.2 and number of rules generated is five after the training procedure. Fig. 8 shows the phase plot of the actual and IRSFNN predicted results for the test patterns.

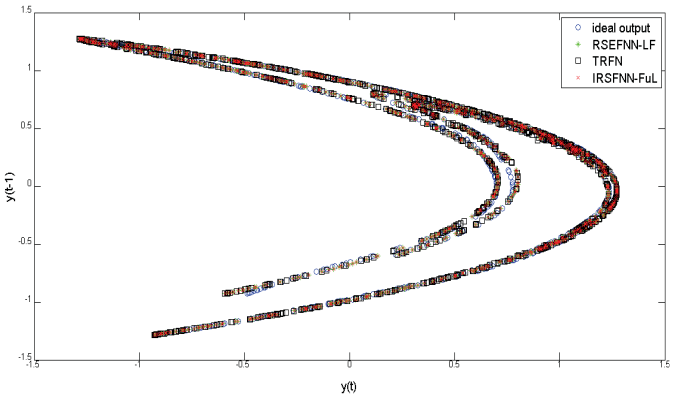

Fig. 8. Results of the phase plot for the chaotic system (blue), RSEFNN_FL (green), TRFN (black), and IRSFNN-FuL (red).

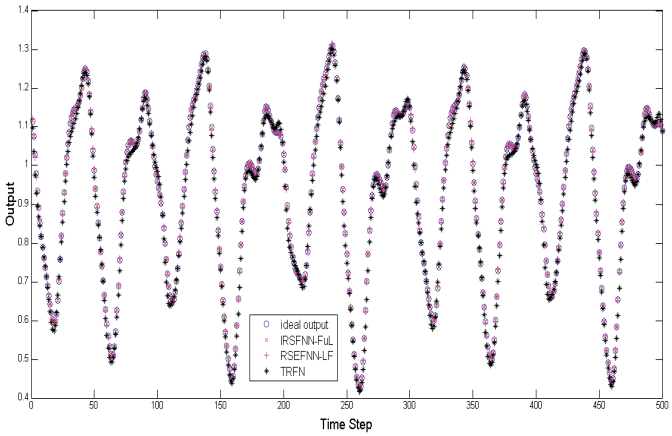

Fig. 9. Test result of chaotic series prediction using IRSFNN-FuL with five rules in Example 4.

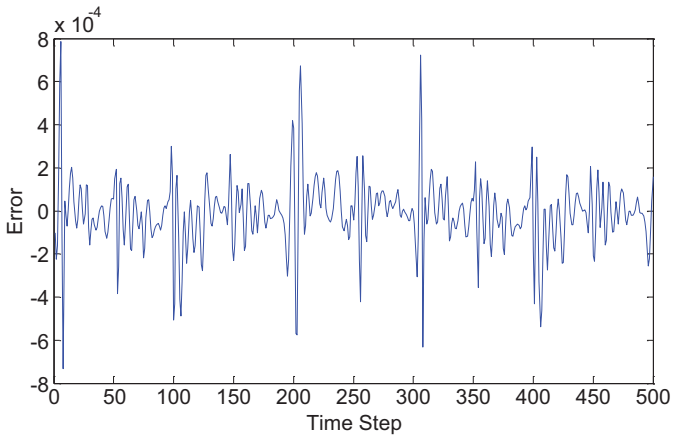

Fig. 10. Prediction errors between the IRSFNN-FuL and actual outputs in Example 4.

Table III includes the network size, parameter numbers, and training and test RMSEs of the IRSFNN. The TSK-type IRSFNN and functional-link-based IRSFNN both use four rules. We find that the latter achieves greater learning performance. In a meaningful comparison, the number of parameters of the IRSFNN must be similar to that of compared models. The compared recurrent models include a recurrent FNN [10], a wavelet recurrent FNN [11], a TSK-TRFN [7], and a recurrent self-evolving FNN with local feedback [14], where the locally recurrent is a simple structure but its performance is superior to that of other compared models. The training epochs, training data, and test data of the compared models are identical to the conditions of the IRSFNN. Table III shows that the IRSFNN exhibits the best performance by using an interactively recurrent structure.

D. Example 4 (Mackey–Glass Chaotic Series Prediction)

The time series prediction problem used in this example is the well-known Mackey–Glass chaotic series. The Mackey–Glass time series is generated from the delay differential equation

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{34}$$

where $\tau = 17$ and the initial value is given as $x(0) = 1.2$. Four past values are used to predict $x(t)$, and the input–output data format is $[x(t-24), x(t-18), x(t-12), x(t-6); x(t)]$. As discussed in [1], [7], [12], [14], 1000 patterns are generated from $t = 124$ to $t = 1123$, with the first 500 patterns being used for training and the remaining 500 for testing. The IRSFNN's training epoch is set to 500, and its structure threshold f_{th} is set to 0.001. After 500 epochs of training procedure, seven rules are generated. The four input dimensions contain 28 fuzzy sets. Fig. 9 displays the prediction results of the IRSFNN-FuL, and Fig. 10 shows the prediction error between desired output and the IRSFNN. Figs. 9 and 10 show an excellent match, suggesting that the proposed scheme in the network indicates an excellent ability to predict the Mackey–Glass time series. Table IV shows the performance, including rules, a total number of parameters, and training and test RMSE for the TSK-type and functional-link-based IRSFNNs.

Table IV lists the performance comparison of the IRSFNN with recently developed fuzzy systems designed by particle swarm algorithms or neural learning ([1], [7], [12], [14], [31]–[37]). Both local linear wavelet NN (LLWNN) [36] and fuzzy wavelet NN (FWNN) [37] employ the wavelet neural network in the consequent part, which has the ability to localize in both time and frequent space.

The compared models with particle swarm algorithm were proposed as a clustering-aided simplex particle swarm optimization (CSPSO) [34], a self-evolving evolutionary learning algorithm (SEELA) designed for neural fuzzy inference system [32], and FLNFN-CCPSO [35]. The FLNFN-CCPSO also uses the functional-link-based neural network to the consequent part in the FLNFN-CCPSO. The proposed models, especially the IRSFNN-FuL, show superior performance to compared models. Although the performance of the IRSFNN-TSK is similar to that of RSEFNN-LF and FWNN, rule number used in the IRSFNN-TSK is fewer than those in RSEFNN-LF and FWNN. The FWNN does not use recurrent structure to memorize previous states and only considers wavelet characteristic in the consequent part to address dynamic systems. Hence, larger rules should be taken by FWNN to obtain good

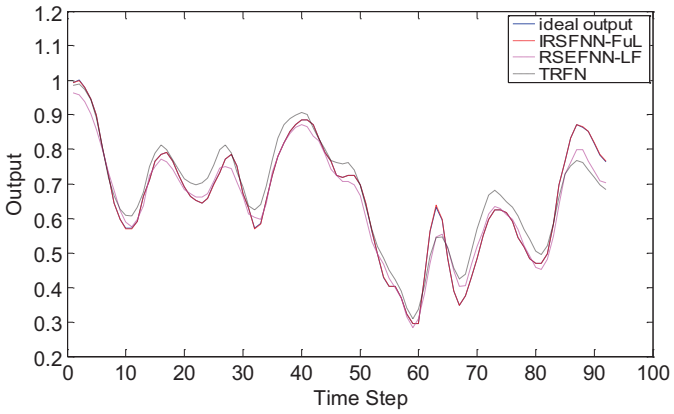

Fig. 11. Test result of Box–Jenkins series using IRSFNN-FuL with three rules in Example 5.

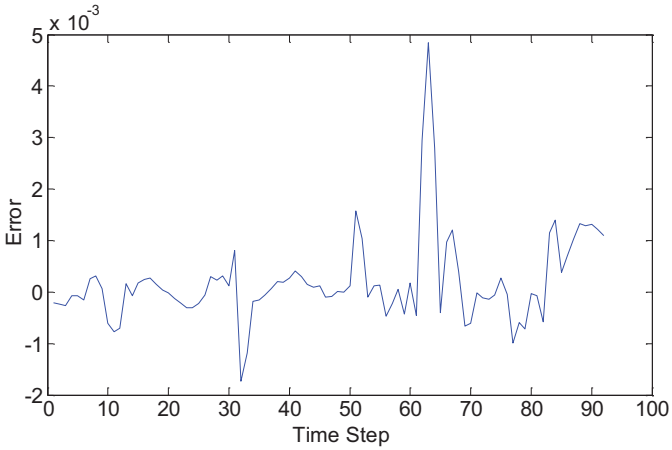

Fig. 12. Prediction errors between the IRSFNN-FuL and actual outputs in Example 5.

performance. Unlike the FWNNs consequent, the consequent of IRSFNN-TSK uses a simple structure of linear combination of input variables. For a fair comparison of using nonlinear system in the consequent part, the test RMSE of IRSFNN-FuL is 11 times lower than that of FWNN. Finally, our proposed IRSFNN-FuL has obtained the best performance among the competitors.

E. Example 5 (Prediction of Box–Jenkins Time Series)

In this example, we consider the use of a real word data set to assess the IRSFNN performance. Much of the literature [31], [36]–[41] uses Box–Jenkins time series to assess the performance of real word data. Box–Jenkins time series data (gas furnace data), which were recorded from a combustion process of a methane-air mixture [36], [42], describe the operation of a gas furnace process with a gas flow rate $u(t)$ and a concentration of $CO_2 y(t)$. To predict the process, $u(t-4)$ and $y(t-1)$ are fed as inputs to the IRSFNN for predicting output $y(t)$. If the appropriate lag number n_u is known in advance, then more past values can be included in the IRSFNN-FuL consequent part for obtaining a greater performance. Therefore, the past value $u(t-3)$, i.e., $n_u = 1$, is used for the IRSFNN-FuL consequent part. The Box–Jenkins time series data provide the 296 available input–output pairs.

For a meaningful comparison, the training samples from the first 200 pairs are used, and the remaining 92 pairs are used for the test samples to predict IRSFNN performance. The structure learning threshold f_{th} is set at 0.01, and the learning factor η is set at 0.08. After 100 training epochs of an IRSFNN-FuL, four rules are generated. The training epoch in the IRSFNN is the same as that in these existing models. As in an IRSFNN, the compared models, except the recurrent ANFIS [31], LLWNN [36], and HyFIS [38], utilize the identical data set which is normalized. The abovementioned three models, recurrent ANFIS, LLWNN, and HyFIS, use identical data set but with a scaled down output to estimate the performance. Hence, they could obtain a lower test RMSE than other compared models. We could assume that the performance of IRSFNN_TSK is superior to that of the above models in terms of the same training and test data set. Fig. 11 shows the prediction results of the IRSFNN. Fig. 12 displays the prediction error between the actual time-series output and the IRSFNN output. As shown in Example 4, Table V lists the parameter numbers, training RMSE, and test RMSE. Table V also shows rules, a total number of network parameters, and training and test error of these compared models, including an HyFIS [38], a LLWNN with hybrid learning (LLWNN+hybrid) [36], a recurrent ANFIS [31], a tree-based neural fuzzy inference system (TNFIS) [39], a Fuzzy neural network (FuNN) [40], a FWNN [37], and TSK-TRFN with supervised learning (TRFN-S) [7]. As can be seen in Table V, the IRSFNN-TSK utilizes fewer rules and achieves a similar performance with the FWNN. For a fair comparison in the consequent part, the test error of IRSFNN-FuL is 31 times lower than that of FWNN. Generally, the results from real world data indicate that the IRSFNN achieves smaller test RMSE than the other compared models.

VI. Conclusion

This paper presented a novel recurrent FNN, called an IRSFNN, for handling problems with time-varying system identification and time series prediction. The proposed IRSFNN approach is effective in modeling dynamic systems because of two major characteristics, online manner and recurrent structure. The online structure learning algorithm enables the network to efficiently identify the required structure of the network and does not need to set any initial IRSFNN structure in advance. The proposed recurrent structure not only effectively stores the local (internal) information but also effectively collects critical global (external) information. The IRSFNN employs the FLNN for the consequent part of its fuzzy rule. In the experiments, the functional-link-based IRSFNN outperformed the TSK-type IRSFNN. The learning algorithm of the variable-dimensional Kalman filter helped to improve network accuracy by tuning the consequent part parameters, and accounted for the change in the network size during learning. This paper demonstrated the consistently superior performance of IRSFNN over several relevant existing systems.

REFERENCES

[1] Y. Gao and M. J. Er, "NARMAX time series model prediction: Feedforward and recurrent fuzzy neural network approaches," *Fuzzy Sets Syst.*, vol. 150, no. 2, pp. 331–350, Mar. 2005.

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

[3] G. C. Mouzouris and J. M. Mendel, "Dynamic nonsingleton fuzzy logic systems for nonlinear modeling," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 2, pp. 199–208, May 1997.

[4] Y. C. Wang, C. J. Chien, and C. C. Teng, "Direct adaptive iterative learning control of nonlinear systems using an output-recurrent fuzzy neural network," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 5, pp. 2144–2154, Jun. 2004.

[5] D. G. Stavrakoudis and J. B. Theocharis, "A recurrent fuzzy neural network for adaptive speech prediction," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Montreal, QC, Canada, Oct. 2007, pp. 2056–2061.

[6] J. B. Theocharis, "A high-order recurrent neuro-fuzzy system with internal dynamics: Application to the adaptive noise cancellation," *Fuzzy Sets Syst.*, vol. 157, no. 4, pp. 471–500, Feb. 2006.

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

[8] C. F. Juang and J. S. Chen, "Water bath temperature control by a recurrent fuzzy controller and its FPGA implementation," *IEEE Trans. Ind. Electron.*, vol. 53, no. 3, pp. 941–949, Jun. 2006.

[9] J. Zhang and A. J. Morris, "Recurrent neuro-fuzzy networks for nonlinear process modeling," *IEEE Trans. Neural Netw.*, vol. 10, no. 2, pp. 313–326, Mar. 1999.

[10] 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, pp. 349–366, Aug. 2000.

[11] C. J. Lin and C. C. Chin, "Prediction and identification using wavelet-based recurrent fuzzy neural networks," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 34, no. 5, pp. 2144–2154, Oct. 2004.

[12] P. A. Mastorocostas and J. B. Theocharis, "A recurrent fuzzy-neural model for dynamic system identification," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 32, no. 2, pp. 176–190, Apr. 2002.

[13] 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, pp. 1597–1612, Dec. 2008.

[14] C. F. Juang, Y. Y. Lin, and C. C. Tu, "A recurrent self-evolving fuzzy neural network with local feedbacks and its application to dynamic system processing," *Fuzzy Sets Syst.*, vol. 161, no. 19, pp. 2552–2568, Oct. 2010.

[15] J. Hu and J. Wang, "Global stability of complex-valued recurrent neural networks with time-delays," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 6, pp. 853–865, Jun. 2012.

[16] F. Ornelas-Tellez, E. N. Sanchez, and A. G. Loukianov, "Discrete-time neural inverse optimal control for nonlinear via passivation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1327–1339, Aug. 2012.

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

[18] C. J. Lin and C. T. Lin, "An ART-based fuzzy adaptive learning control network," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 4, pp. 477–496, Nov. 1997.

[19] W. S. Lin, C. H. Tsai, and J. S. Liu, "Robust neuro-fuzzy control of multivariable systems by tuning consequent membership functions," *Fuzzy Sets Syst.*, vol. 124, no. 2, pp. 181–195, Dec. 2001.

[20] J. S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, Jun. 1993.

[21] C. F. Juang and C. T. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 1, pp. 12–31, Feb. 1998.

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

[23] J. C. Patra, R. N. Pal, and B. N. Chatterji, and G. Panda, "Identification of nonlinear dynamic systems using functional link artificial neural networks," *IEEE Trans Syst., Man, Cybern.*, vol. 29, no. 2, pp. 254–262, Apr. 1999.

[24] M. Alci, "Fuzzy rule-base driven orthogonal approximation," *Neural Comp. Appl.*, vol. 17, nos. 5–6, pp. 501–507, 2008.

[25] M. Alci, "Fuzzy systems on orthogonal bases," in *Proc. Dynamics Continuous Discrete Impuls. Syst.-Series Math. Anal.*, 2007, pp. 671–676.

[26] M. Alci and S. Beyhan, "Trigonometric functions based neural networks for system identification," in *Proc. 5th IFAC Intl.*, May 2007, pp. 141–144.

[27] S. Beyhan and M. Alci, "An orthogonal ARX network for identification and control of nonlinear systems," in *Proc. XXII Int. Symp. Inf., Commun. Autom. Technol.*, Oct. 2009, pp. 1–5.

[28] J. Y. Chang, Y. Y. Lin, M. F. Han, and C. T. Lin, "A functional-link based Interval Type-2 Compensatory Fuzzy Neural Network for Nonlinear System modeling," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Jun. 2011, pp. 939–943.

[29] C. F. Juang, R. B. Huang, and Y. Y. Lin, "A recurrent self-evolving interval type-2 fuzzy neural network for dynamic system processing," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 5, pp. 1092–1105, Oct. 2009.

[30] G. Chen, Y. Chen, and H. Ogmen, "Identifying chaotic system via a wiener-type cascade model," *IEEE Trans. Contr. Syst.*, vol. 17, no. 5, pp. 29–36, Oct. 1997.

[31] H. Tamura, K. Tanno, H. Tanaka, C. Vairappan, and Z. Tang, "Recurrent type ANFIS using local search technique for time series prediction," in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, Dec. 2008, pp. 380–383.

[32] C. J. Lin, C. H. Chen, and C. T. Lin, "Efficient self-evolving evolutionary learning for neurofuzzy inference systems," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 1476–1490, Dec. 2008.

[33] S. Paul and S. Kumar, "Subsethood-product fuzzy neural inference system (SuPFuNIS)," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 578–599, May 2002.

[34] C. F. Juang, C. H. Hsu, and I. F. Chung, "Automatic construction of feedforward/recurrent fuzzy systems by clustering-aided simplex particle swarm optimization," *Fuzzy Sets Syst.*, vol. 158, no. 18, pp. 1979–1996, Sep. 2007.

[35] C. J. Lin, C. H. Chen, and C. T. Lin, "A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications," *IEEE Trans. Syst., Man, Cybern., C, Appl. Rev.*, vol. 39, no. 1, pp. 55–68, Jan. 2009.

[36] Y. Chen, B. Yang, and J. Dong, "Time-series prediction using a local linear wavelet neural network," *Neurocomput.*, vol. 69, nos. 4–6, pp. 449–465, Jan. 2006.

[37] S. Yilmaz and Y. Oysal, "Fuzzy wavelet neural network models for prediction and identification of dynamic system," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1599–1609, Oct. 2010.

[38] J. Kim and N. Kasabov, "HyFIS: Adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems," *Neural Netw.*, vol. 12, no. 9, pp. 1301–1319, Nov. 1999.

[39] E. Y. Cheu, H. C. Quek, and S. K. Ng, "TNFIS: Tree-based neural fuzzy inference system," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jun. 2008, pp. 398–405.

[40] N. K. Kasabov, J. Kim, M. J. Watts, and A. R. Gray, "FuNN/2-Afuzzy neural network architecture for adaptive learning and knowledgeacquisition," *Inform. Sci.*, vol. 101, nos. 3–4, pp. 155–175, Oct. 1997.

[41] M. Alci and M. H. Asyali, "Nonlinear system identification via Laguerre network based fuzzy system," *Fuzzy Sets Syst.*, vol. 160, no. 24, pp. 3518–3529, Dec. 2009.

[42] G. E. P. Box, *Time Series Analysis, Forecasting and Control*, San Francisco, CA: Holden, 1970.

Yang-Yin Lin received the B.S. degree from the Department of Electronics Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan, in 2005, and the M.S. degree from the Institute of Electrical Engineering, National Chung-Hsing University, Taichung, Taiwan, in 2008. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, National Chiao-Tung University, Hsinchu, Taiwan.

His current research interests include computational intelligence, type-2 fuzzy neural network and FPGA chip design.

Jyh-Yeong Chang (S'84–M'86) received the B.S. degree in control engineering and the M.S. degree in electronic engineering in 1980, both from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 1976 and 1980, respectively, and the Ph.D. degree in electrical engineering from North Carolina State University, Raleigh, in 1987.

He was a Research Fellow with the Chung Shan Institute of Science and Technology, Lung-Tan, Taiwan, from 1976 to 1978 and from 1980 to 1982. In 1987, he was an Associate Professor with the Department of Electrical and Control Engineering, NCTU, where he is currently a Professor. His current research interests include neural fuzzy systems, video processing, surveillance, and bioinformatics.

Chin-Teng Lin (S'88–M'91–SM'99–F'05) received the B.S. degree from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 1986, and the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1989 and 1992, respectively. He is currently the Provost, Chair Professor of electrical and computer engineering, and the Director of Brain Research Center, National Chiao Tung University. He has published more than 120 journal papers in neural networks, fuzzy systems, multimedia hardware/software, and cognitive neuro-engineering, including approximately 74 IEEE journal papers.

Dr. Lin was invited to be an IEEE fellow for his contributions to biologically inspired information systems in 2005. He was elected as the Editor-in-chief of IEEE TRANSACTIONS ON FUZZY SYSTEMS. He also served on the Board of Governors at the IEEE Circuits and Systems (CAS) Society from 2005 to 2008, the IEEE Systems, Man, Cybernetics (SMC) Society from 2003 to 2005, the IEEE Computational Intelligence Society from 2008 to 2010, and the Chair of the IEEE Taipei Section from 2009 to 2010. He served as the Deputy Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II from 2006 to 2008. He is the co-author of a book entitled *Neural Fuzzy Systems* (Prentice-Hall) and the author of a book entitled *Neural Fuzzy Control Systems with Structure and Parameter Learning.*