# A Recurrent Self-Evolving Interval Type-2 Fuzzy Neural Network for Dynamic System Processing

Chia-Feng Juang, *Senior Member, IEEE*, Ren-Bo Huang, and Yang-Yin Lin

***Abstract*—This paper proposes a recurrent self-evolving interval type-2 fuzzy neural network (RSEIT2FNN) for dynamic system processing. An RSEIT2FNN incorporates type-2 fuzzy sets in a recurrent neural fuzzy system in order to increase the noise resistance of a system. The antecedent parts in each recurrent fuzzy rule in the RSEIT2FNN are interval type-2 fuzzy sets, and the consequent part is of the Takagi–Sugeno–Kang (TSK) type with interval weights. The antecedent part of RSEIT2FNN forms a local internal feedback loop by feeding the rule firing strength of each rule back to itself. The TSK-type consequent part is a linear model of exogenous inputs. The RSEIT2FNN initially contains no rules; all rules are learned online via structure and parameter learning. The structure learning uses online type-2 fuzzy clustering. For the parameter learning, the consequent part parameters are tuned by a rule-ordered Kalman filter algorithm to improve learning performance. The antecedent type-2 fuzzy sets and internal feedback loop weights are learned by a gradient descent algorithm. The RSEIT2FNN is applied to simulations of dynamic system identifications and chaotic signal prediction under both noise-free and noisy conditions. Comparisons with type-1 recurrent fuzzy neural networks validate the performance of the RSEIT2FNN.**

***Index Terms*—Dynamic system identification, online fuzzy clustering, recurrent fuzzy neural networks (RFNNs), recurrent fuzzy systems, type-2 fuzzy systems.**

## I. Introduction

THE TOPOLOGIES of recurrent networks include feedback loops, which are used to memorize past information. In contrast with pure feedforward architectures, which exhibit static input–output behavior, recurrent networks are able to store information from the past (e.g., prior system states) and are, thus, more appropriate for the analysis of dynamic systems. Some recurrent fuzzy neural networks (RFNNs) have already been proposed [1]–[9] to deal with temporal characteristic problems, and have been shown to outperform feedforward FNNs and recurrent NNs. One category of RFNNs uses feedback loops from the network output(s) as a recurrence structure [1]–[3]. The authors in [1] and [3] proposed an output RFNN where the output values are fed back as input values. A recurrent neural fuzzy network is proposed in [2], where the consequent of a rule is a reduced linear model in autoregressive form with exogenous inputs. Another category of RFNNs uses feedback loops from internal state variables as its recurrence structure [4]–[9]. The recurrence property in studies [4] and [5] is achieved by feeding the output of each membership function (MF) back to itself, and therefore, each membership value is influenced by and only by its past values. Recurrent self-organizing neural fuzzy inference networks (RSONFINs) [6] and Takagi–Sugeno–Kang (TSK) type recurrent fuzzy networks (TRFN) [7], [9] use a global feedback structure, where the firing strengths of each rule are summed and fed back as internal network inputs.

All of the aforementioned RFNNs use type-1 fuzzy sets. In recent years, studies on type-2 fuzzy logic systems (FLSs) have drawn much attention [10]–[14]. Type-2 FLSs are extensions of type-1 FLS, where the membership value of a type-2 fuzzy set is a type-1 fuzzy number. Type-2 FLSs appear to be a more promising method than their type-1 counterparts in handling problems with uncertainties, and have already been successfully applied in several situations [15]–[18]. Some interval type-2 FNNs have been proposed for the automatic design of interval type-2 FLS [19]–[23]. A gradient descent algorithm was proposed for interval type-2 FNN learning in [19]–[22]. In the center-of-sets type reduction process, the consequent values in the interval type-2 FLS are rearranged in ascending order to compute the interval outputs using the Karnik–Mendel iterative procedure [19]–[23]. During the parameter learning process, the consequent values change, and their ascending orders and corresponding fuzzy rule orders should change accordingly. The parameter learning equations in [19], [20], [22], and [23] did not explicitly address this fuzzy rule reordering problem. Parameter learning equations that considered the fuzzy rule reordering problem using a gradient descent algorithm were proposed in [21]. This paper proposes a rule-ordered Kalman filter algorithm for recurrent type-2 FNN consequent part parameter learning. The algorithm derives detailed learning equations, taking into account the rule-reordering problem in network output computation.

This paper proposes a recurrent type-2 FNN, i.e., the recurrent self-evolving interval type-2 FNN (RSEIT2FNN), for dynamic system processing. The self-evolving property means that the RSEIT2FNN can automatically evolve its network structure and parameters according to training data, i.e., the task of preassigning network structure (rule numbers and initial fuzzy set shapes) is no longer necessary. The major contributions of the RSEIT2FNN are twofold. The first contribution is the proposal of a novel RFNN structure with the introduction of type-2 fuzzy sets. Current studies on type-2 FNNs only focus on feedforward network structures to handle static input–output

Manuscript received June 30, 2008; revised October 20, 2008, January 12, 2009, and March 20, 2009; accepted March 25, 2009. First published May 2, 2009; current version published October 8, 2009. This work was supported by the Ministry of Education, Taiwan, under the Aiming for Top University plan.

C.-F. Juang and R.-B. Huang are with the Department of Electrical Engineering, National Chung-Hsing University, Taichung 402, Taiwan (e-mail: cfjuang@dragon.nchu.edu.tw; g9664202@mail.nchu.edu.tw).

Y.-Y. Lin is with the Department of Electrical Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan (e-mail: daviddavid715288@yahoo.com.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/TFUZZ.2009.2021953

mapping problems. In the RSEIT2FNN structure, local feedback loops in the antecedent part are formed by feeding the rule firing strength of each rule back to itself, and the consequent part is a combination of current and lagged network inputs. The second contribution is the proposal of novel structure and parameter learning algorithms to improve the network learning performance. The aforementioned type-2 FNNs learn only parameters, where the structures are all fixed and must be assigned in advance. The RSEIT2FNN learns both the structure and parameters concurrently and online. All of the rules in an RSEIT2FNN are generated online. The consequent parameters in the RSEIT2FNN are learned by a rule-ordered Kalman filter algorithm to improve learning performance. The antecedent part parameters and rule feedback weights are learned by a gradient descent learning algorithm. Several simulations are conducted to verify RSEIT2FNN performance. These simulations also compare the RSEIT2FNN with recurrent type-1 FNNs, feedforward type-1 FNNs, and other type-2 FNNs.

The rest of this paper is organized as follows. Section II introduces the RSEIT2FNN structure. Section III introduces the structure and parameter learning methods in an RSEIT2FNN. Section IV simulates three examples on dynamic systems identification and chaotic series prediction. Finally, Section V draws conclusions.

## II. RSEIT2FNN Structure

This section introduces the structure of an RSEIT2FNN. Suppose that the dynamic system to be processed is a multi-input–multioutput (MIMO) system that consists of $n_u$ control inputs and $n_o$ outputs and that the control input and dynamic system output vectors are denoted by $\boldsymbol{u} = (u_1, \ldots, u_{n_u})$ and $\boldsymbol{y}_p = (y_{p1}, \ldots, y_{pn_O})$, respectively, where $n_u$ and $n_o$ denote the input and output dimensions, respectively. Fig. 1 shows the proposed MIMO six-layered RSEIT2FNN structure. The consequent of each recurrent fuzzy rule is of first-order Takagi–Sugeno-Kang (TSK) type and executes a linear function. The detailed mathematical functions of each layer are introduced next.

*1) Layer 1 (Input Layer):* The inputs are crisp values. Only the current states $\boldsymbol{x}(t) = (\boldsymbol{u}(t), \boldsymbol{y}_p(t))$ are fed as inputs to this layer, in contrast to feedforward FNNs where both current and past states are fed as inputs to the input layer. To unify the input range, each node in this layer scales the inputs to lie in the range around $[-1,\ 1]$. Note that there are no weights to be adjusted in this layer.

*2) Layer 2 (MF Layer):* Each node in this layer defines an interval type-2 MF. For the $i$th interval type-2 fuzzy set $\tilde{A}_j^i$ in input variable $x_j$, $j = 1, \ldots, n_u + n_o$, two types of MFs are studied. For the first type, we use a Gaussian primary MF having a fixed standard deviation (STD) $\sigma_j^i$ and an uncertain mean that takes on values in $[m_{j1}^i, m_{j2}^i]$ [see Fig. 2(a)], i.e.,

$$\mu_{\tilde{A}_j^i} = \exp\left\{-\frac{1}{2}\left(\frac{x_j - m_j^i}{\sigma_j^i}\right)^2\right\} \equiv N(m_j^i, \sigma_j^i; x_j)$$
$$m_j^i \in [m_{j1}^i, m_{j2}^i]. \quad (1)$$

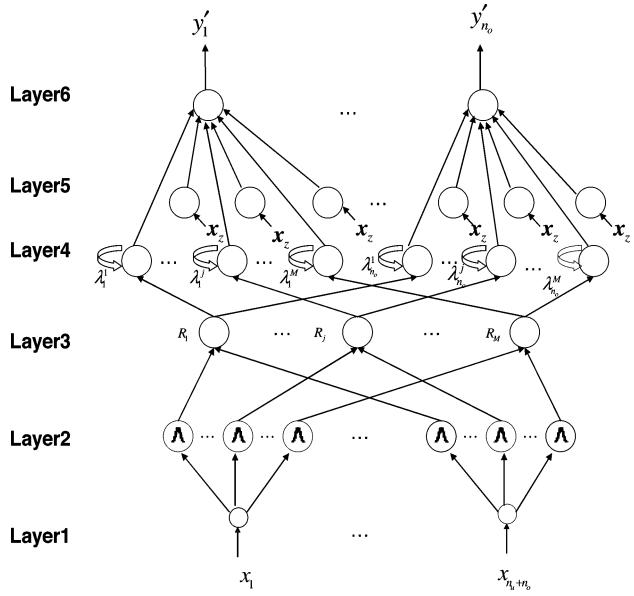



Fig. 1. Structure of the RSEIT2FNN, where each node in layer 4 forms an internal feedback loop and each node in layer 5 functions as a linear combination of current and lagged network inputs (denoted as $\boldsymbol{x}_z$).

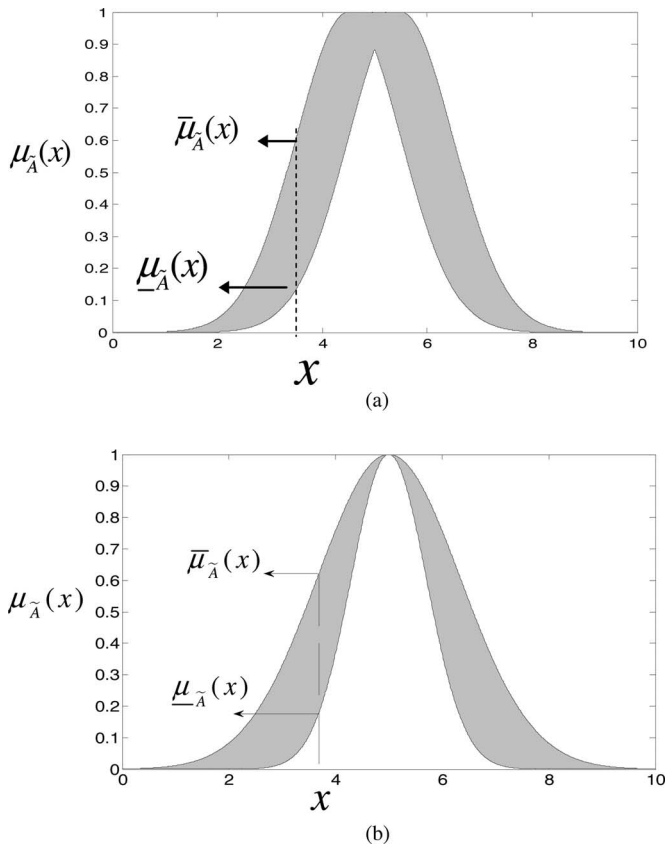



Fig. 2. Interval type-2 fuzzy set. (a) Uncertain mean. (b) Uncertain STD.

An RSEIT2FNN using this type of MF is called RSEIT2FNN-UM. The footprint of uncertainty (FOU) of this MF can be represented as a bounded interval in terms of its upper MF $\overline{\mu}_j^i$

and lower MF $\underline{\mu}_j^i$, where

$$\overline{\mu}_j^i(x_j) = \begin{cases} N(m_{j1}^i, \sigma_j^i; x_j), & x_j < m_{j1}^i \\ 1, & m_{j1}^i \le x_j \le m_{j2}^i \\ N(m_{j2}^i, \sigma_j^i; x_j), & x_j > m_{j2}^i \end{cases} \tag{2}$$

and

$$\underline{\mu}_j^i(x_j) = \begin{cases} N(m_{j2}^i, \sigma_j^i; x_j), & x_j \le \dfrac{m_{j1}^i + m_{j2}^i}{2} \\ N(m_{j1}^i, \sigma_j^i; x_j), & x_j > \dfrac{m_{j1}^i + m_{j2}^i}{2} \end{cases} \tag{3}$$

i.e., the output $\mu_{\tilde{A}_j^i}$ of each node can be represented as an interval $[\underline{\mu}_j^i, \overline{\mu}_j^i]$. For the second type, we use a Gaussian primary MF having a fixed mean $m_j^i$ and an uncertain STD that takes on values in $[\sigma_{j1}^i, \sigma_{j2}^i]$. An RSEIT2FNN using this type of MF is called RSEIT2FNN-UD. The membership value is $\mu_{\tilde{A}_j^i} = [\underline{\mu}_j^i, \overline{\mu}_j^i]$, where

$$\overline{\mu}_j^i(x_j) = N(m_j^i, \sigma_{j2}^i; x_j) \tag{4}$$

and

$$\underline{\mu}_j^i(x_j) = N(m_j^i, \sigma_{j1}^i; x_j). \tag{5}$$

The two types of interval type-2 MFs mentioned before are also widely used in many previous studies [19]–[25].

*3) Layer 3 (Spatial Firing Layer):* Each node in this layer corresponds to one fuzzy rule and functions as a spatial rule node. Each node performs a fuzzy meet operation on inputs from layer 2 using an algebraic product operation to obtain a spatial firing strength $F^i$. The spatial firing strength is an interval type-1 fuzzy set and is computed as follows [26]:

$$F^i = [\underline{f}^i, \overline{f}^i], \qquad i = 1, \ldots, M \tag{6}$$

where

$$\overline{f}^i = \prod_{j=1}^{n_u+n_o} \overline{\mu}_j^i, \qquad \underline{f}^i = \prod_{j=1}^{n_u+n_o} \underline{\mu}_j^i \tag{7}$$

and $M$ is the total number of rules.

*4) Layer 4 (Temporal Firing Layer):* Each node in this layer is a recurrent rule node that forms an internal feedback loop. The output of a recurrent rule node is a temporal firing strength that depends not only on the current spatial firing strength but on the previous temporal firing strength as well. The temporal firing strength $\Psi_q^i(t) = [\overline{\psi}_q^i(t), \underline{\psi}_q^i(t)]$, $i = 1, \ldots, M$ and $q = 1, \ldots, n_o$ is an interval given as a linear combination of the spatial firing strength $F^i(t)$ and the last temporal firing strength $\Psi_q^i(t-1)$ by the equation

$$\Psi_q^i(t) = \lambda_q^i \cdot F^i(t) + (1 - \lambda_q^i)\Psi_q^i(t-1) \tag{8}$$

where $\lambda_q^i$ is a feedback weight and $0 \le \lambda_q^i \le 1$. Equation (8) may be written as

$$[\overline{\psi}_q^i(t), \underline{\psi}_q^i(t)] = \lambda_q^i[\overline{f}^i(t), \underline{f}^i(t)] + (1-\lambda_q^i)[\overline{\psi}_q^i(t-1), \underline{\psi}_q^i(t-1)] \tag{9}$$

where

$$\overline{\psi}_q^i(t) = \lambda_q^i \overline{f}^i(t) + (1 - \lambda_q^i)\overline{\psi}_q^i(t-1) \tag{10}$$

and

$$\underline{\psi}_q^i(t) = \lambda_q^i \underline{f}^i(t) + (1 - \lambda_q^i)\underline{\psi}_q^i(t-1). \tag{11}$$

*5) Layer 5 (Consequent Layer):* Each node in this layer is called a consequent node and functions as a linear model with exogenous inputs and time-delay synapses. Each recurrent rule node in layer 4 has a corresponding consequent node in layer 5. The linear model is a linear combination of the current input states $\boldsymbol{x}(t) = (\boldsymbol{u}(t), \boldsymbol{y}_p(t)) = (u_1(t), \ldots, u_{n_u}(t), y_{p1}(t), \ldots, y_{pn_o}(t))$, together with their lagged values. The output $\tilde{y}_q^i(t+1)$, $i = 1, \ldots, M$ and $q = 1, \ldots, n_o$, of the $i$th consequent node connecting to the $q$th network output variable is computed as follows:

$$\tilde{y}_q^i(t+1) = \sum_{j=0}^{n_u}\sum_{k=0}^{N_j} \tilde{a}_{jkq}^i u_j(t-k) + \sum_{j=1}^{n_o}\sum_{k=0}^{O_j} \tilde{a}_{(j+n_u)kq}^i y_{pj}(t-k) \tag{12}$$

where $u_0(t) \triangleq 1$ and $N_0 \triangleq 0$, $N_j$ and $O_j$ are the maximum lag numbers of the control input $u_j(t)$ and the system output $y_{pj}(t)$, respectively, and $\tilde{a}_{jkq}^i$ are interval sets denoted by

$$\tilde{a}_{jkq}^i = [c_{jkq}^i - s_{jkq}^i, c_{jkq}^i + s_{jkq}^i] \tag{13}$$

where $c_{jkq}^i$ and $s_{jkq}^i$ denote the center and spread, respectively, of the interval. The inclusion of the lagged values of $\boldsymbol{u}(t)$ and $\boldsymbol{y}_p(t)$ in the linear consequent part instead of the antecedent part simplifies the network computation process for dynamic system processing, especially when interval type-2 fuzzy sets are used. The output $\tilde{y}_q^i(t+1)$ is an interval type-1 set, which is denoted by $[\tilde{y}_{lq}^i, \tilde{y}_{rq}^i]$, where the indices $l$ and $r$ denote left and right limits, respectively. According to (12) and (13), the node output is given as

$$\begin{aligned} \tilde{y}_q^i = [\tilde{y}_{lq}^i, \tilde{y}_{rq}^i] &= \sum_{j=0}^{n_u}\sum_{k=0}^{N_j} [c_{jkq}^i - s_{jkq}^i, c_{jkq}^i + s_{jkq}^i] u_j(t-k) \\ &+ \sum_{j=1}^{n_o}\sum_{k=0}^{O_j} [c_{(j+n_u)kq}^i - s_{(j+n_u)kq}^i, c_{(j+n_u)kq}^i \\ &+ s_{(j+n_u)kq}^i] y_{pj}(t-k) \end{aligned} \tag{14}$$

i.e.,

$$\begin{aligned} \tilde{y}_{lq}^i &= \sum_{j=0}^{n_u}\sum_{k=0}^{N_j} c_{jkq}^i u_j(t-k) + \sum_{j=1}^{n_o}\sum_{k=0}^{O_j} c_{(j+n_u)kq}^i y_{pj}(t-k) \\ &- \sum_{j=0}^{n_u}\sum_{k=0}^{N_j} s_{jkq}^i |u_j(t-k)| - \sum_{j=1}^{n_o}\sum_{k=0}^{O_j} s_{(j+n_u)kq}^i |y_{pj}(t-k)| \end{aligned} \tag{15}$$

and

$$\tilde{y}_{rq}^{i} = \sum_{j=0}^{n_u}\sum_{k=0}^{N_j} c_{jkq}^{i} u_j(t-k) + \sum_{j=1}^{n_o}\sum_{k=0}^{O_j} c_{(j+n_u)kq}^{i} y_{pj}(t-k) + \sum_{j=0}^{n_u}\sum_{k=0}^{N_j} s_{jkq}^{i}|u_j(t-k)| + \sum_{j=1}^{n_o}\sum_{k=0}^{O_j} s_{(j+n_u)kq}^{i}|y_{pj}(t-k)|. \tag{16}$$

*6) Layer 6 (Output Layer):* Each node in this layer corresponds to one output variable. The $q$th output layer node computes the network output variable $y_q'$ using type reduction followed by defuzzification operations. In the type reduction, the type-reduced set is an interval type-1 fuzzy set $[y_{lq}', y_{rq}']$. The outputs $y_{lq}'$ and $y_{rq}'$ can be computed using the Karnik–Mendel iterative procedure [11]. In this procedure, the consequent parameters are reordered in ascending order. Let $\tilde{\boldsymbol{y}}_{lq} = (\tilde{y}_{lq}^1, \ldots, \tilde{y}_{lq}^M)$ and $\tilde{\boldsymbol{y}}_{rq} = (\tilde{y}_{rq}^1, \ldots, \tilde{y}_{rq}^M)$ denote the original rule-ordered consequent values, and let $\hat{\boldsymbol{y}}_{lq} = (\hat{y}_{lq}^1, \ldots, \hat{y}_{lq}^M)$ and $\hat{\boldsymbol{y}}_{rq} = (\hat{y}_{rq}^1, \ldots, \hat{y}_{rq}^M)$ denote the reordered sequences, where $\hat{y}_{lq}^1 \le \hat{y}_{lq}^2 \le \cdots \le \hat{y}_{lq}^M$ and $\hat{y}_{rq}^1 \le \hat{y}_{rq}^2 \le \cdots \le \hat{y}_{rq}^M$. The relationship between $\tilde{\boldsymbol{y}}_{lq}$, $\tilde{\boldsymbol{y}}_{rq}$, $\hat{\boldsymbol{y}}_{lq}$, and $\hat{\boldsymbol{y}}_{rq}$ is

$$\hat{\boldsymbol{y}}_{lq} = Q_l \tilde{\boldsymbol{y}}_{lq} \qquad \text{and} \qquad \hat{\boldsymbol{y}}_{rq} = Q_r \tilde{\boldsymbol{y}}_{rq} \tag{17}$$

where $Q_l$ and $Q_r$ are $M \times M$ permutation matrices with elementary vectors (i.e., vectors all of whose elements are zero except one element that is equal to one) as columns, and these vectors are arranged (permuted) to move elements in $\tilde{\boldsymbol{y}}_{lq}$ and $\tilde{\boldsymbol{y}}_{rq}$ to new locations in ascending order in the transformed vectors $\hat{\boldsymbol{y}}_{lq}$ and $\hat{\boldsymbol{y}}_{rq}$, respectively. The original rule firing strength orders $\underline{\psi} = (\underline{\psi}_q^1, \underline{\psi}_q^2, \ldots, \underline{\psi}_q^M)^T$ and $\overline{\psi} = (\overline{\psi}_q^1, \overline{\psi}_q^2, \ldots, \overline{\psi}_q^M)^T$ are reordered accordingly. To compute the output $y_{lq}'$, the new rule orders for $\underline{\psi}$ and $\overline{\psi}$ are $Q_l\underline{\psi}$ and $Q_l\overline{\psi}$, respectively. To compute the output $y_{rq}'$, the new rule orders for $\underline{\psi}$ and $\overline{\psi}$ are $Q_r\underline{\psi}$ and $Q_r\overline{\psi}$, respectively. According to [21], the output $y_{lq}'$ can be computed as

$$y_{lq}' = \frac{\sum_{i=1}^{L}(Q_l\overline{\psi})_i \hat{y}_{lq}^i + \sum_{i=L+1}^{M}(Q_l\underline{\psi})_i \hat{y}_{lq}^i}{\sum_{i=1}^{L}(Q_l\overline{\psi})_i + \sum_{i=L+1}^{M}(Q_l\underline{\psi})_i} = \frac{\overline{\psi}^{\mathrm{T}} Q_l^T E_1^T E_1 Q_l \tilde{\boldsymbol{y}}_{lq} + \underline{\psi}^T Q_l^T E_2^T E_2 Q_l \tilde{\boldsymbol{y}}_{lq}}{\boldsymbol{p}_l^T Q_l \overline{\psi} + \boldsymbol{g}_l^T Q_l \underline{\psi}} \tag{18}$$

where $L$ and $R$ denote the left and right crossover points, respectively

$$\boldsymbol{p}_l \triangleq (\underbrace{1, 1, \ldots, 1}_{L}, 0, \ldots 0)^T \in \Re^{M\times 1}$$
$$\boldsymbol{g}_l = (0, \ldots, 0, \underbrace{1, \ldots, 1}_{M-L})^T \in \Re^{M\times 1} \tag{19}$$
$$E_1 = (\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_L, \boldsymbol{0}, \ldots, \boldsymbol{0}) \in \Re^{L\times M}$$
$$E_2 = (\boldsymbol{0}, \ldots, \boldsymbol{0}, \boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \ldots, \boldsymbol{\varepsilon}_{M-L}) \in \Re^{(M-L)\times M} \tag{20}$$

and where $\boldsymbol{e}_i \in \Re^{L\times 1}$ and $\boldsymbol{\varepsilon}_i \in \Re^{M-L}$ are elementary vectors. Similarly, the output $y_{rq}'$ can be computed as

$$y_{rq}' = \frac{\sum_{i=1}^{R}(Q_r\underline{\psi})_i \hat{y}_{rq}^i + \sum_{i=R+1}^{M}(Q_r\overline{\psi})_i \hat{y}_{rq}^i}{\sum_{i=1}^{R}(Q_r\underline{\psi})_i + \sum_{i=R+1}^{M}(Q_r\overline{\psi})_i} = \frac{\underline{\psi}^T Q_r^T E_3^T E_3 Q_r \tilde{y}_{rq} + \overline{\psi}^T Q_r^T E_4^T E_4 Q_r \tilde{\boldsymbol{y}}_{rq}}{\boldsymbol{p}_r^T Q_r \underline{\psi} + \boldsymbol{g}_r^T Q_r \overline{\psi}} \tag{21}$$

where

$$\boldsymbol{p}_r \triangleq (\underbrace{1, 1, \ldots, 1}_{R}, 0, \ldots 0)^T \in \Re^{M\times 1}$$
$$\boldsymbol{g}_r = (0, \ldots, 0, \underbrace{1, \ldots, 1}_{M-R})^T \in \Re^{M\times 1} \tag{22}$$
$$E_3 = (\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_R, \boldsymbol{0}, \ldots, \boldsymbol{0}) \in \Re^{R\times M}$$
$$E_4 = (\boldsymbol{0}, \ldots, \boldsymbol{0}, \boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \ldots, \boldsymbol{\varepsilon}_{M-R}) \in \Re^{(M-R)\times M} \tag{23}$$

and where $\boldsymbol{e}_i \in \Re^{R\times 1}$ and $\boldsymbol{\varepsilon}_i \in \Re^{(M-R)\times 1}$ are elementary vectors. In contrast to the prior studies [19], [20], [22], [23], the outputs $y_{lq}'$ and $y_{rq}'$ in (18) and (21) are expressed in the original rule-ordered format. This expression is helpful in deriving the proposed parameter learning algorithm discussed in Section III-B. Finally, the defuzzification operation defuzzifies the interval set $[y_{lq}', y_{rq}']$ by computing the average of $y_{lq}'$ and $y_{rq}'$. Hence, the defuzzified output for network output variable $y_q'$ is

$$y_q' = \frac{y_{lq}' + y_{rq}'}{2}. \tag{24}$$

## III. RSEIT2FNN Learning

The RSEIT2FNN evolves all of its composing recurrent type-2 fuzzy rules by simultaneous structure and parameter learning. The following sections introduce more details on the structure and parameter learning algorithms.

### A. Structure Learning

The structure learning algorithm is responsible for online rule generation. A previous study [28] used the rule firing strength as a criterion for type-1 fuzzy rule generation. This idea is extended to type-2 fuzzy rule generation criteria in an RSEIT2FNN. The spatial firing strength $F^i$ in (6) is used as the criterion to determine whether a rule should be generated. Since this firing strength is an interval, its center is

$$f_c^i = \frac{1}{2}(\overline{f}^i + \underline{f}_i). \tag{25}$$

The spatial firing strength center then serves as a rule generation criterion.

Structure learning of an RSEIT2FNN-UM is introduced as follows. For the first incoming piece of data $x$, a new rule is generated, with the uncertain mean and center of each new type-2 fuzzy set assigned by

$$[m_{j1}^1, m_{j2}^1] = [x_j - 0.1, x_j + 0.1] \qquad \text{and} \qquad \sigma_j^i = \sigma_{\text{fixed}} \qquad j = 1, \ldots, n_u + n_o \tag{26}$$

where $\sigma_{\text{fixed}}$ is a predefined threshold ($\sigma_{\text{fixed}} = 0.3$ in this paper) that determines the fuzzy set width. For each subsequent piece of incoming data $\boldsymbol{x}(t)$, find

$$I = \arg \max_{1 \le i \le M(t)} f_c^i(\vec{x}) \tag{27}$$

where $M(t)$ is the number of existing rules at time $t$. If $f_c^I(\vec{x}) \le f_{\text{th}}$, for some prespecified threshold $f_{\text{th}} \in (0, 1)$, then a new rule is generated. Once a new rule is generated, the initial uncertain means and widths of the corresponding new type-2 fuzzy sets are assigned as

$$[m_{j1}^{M(t)+1}, m_{j2}^{M(t)+1}] = [x_j(t) - 0.1, x_j(t) + 0.1] \tag{28}$$

and

$$\sigma_j^{M(t)+1} = \beta \left( \sum_{j=1}^{n_u+n_o} \left( x_j - \left( \frac{m_{j1}^J + m_{j2}^J}{2} \right) \right)^2 \right)^{0.5}. \tag{29}$$

The network inputs are scaled to lie in the range $[-1, 1]$ in layer 1. Equations (26) and (28) set an initial uncertain mean range of 0.2 according to this input range. If the uncertain mean range is too small, then the initial type-2 fuzzy set becomes too close to a type-1 fuzzy set. In contrast, if this range is too large, then the uncertain mean covers most of the input range. Equation (29) indicates that the initial width is equal to the Euclidean distance between the current input data point $\boldsymbol{x}$ and its nearest cluster mean average times an overlapping parameter $\beta$. In this paper, we set $\beta = 0.5$ so that the width of the new fuzzy set is half the Euclidean distance, and a suitable overlapping between clusters is generated.

For structure learning of an RSEIT2FNN-UD, the learning process is similar to RSEIT2FNN-UM, except that (26), (28), and (29) are slightly modified. Equation (26) is changed to be

$$m_j^1 = x_j \quad \text{and} \quad [\sigma_{j1}^1, \sigma_{j2}^1] = [\sigma_{\text{fixed}}, \sigma_{\text{fixed}} + 0.1] \quad j = 1, \ldots, n_u + n_o. \tag{30}$$

Equations (28) and (29) are changed to be

$$m_j^{M(t)+1} = x_j(t) \tag{31}$$

and

$$[\sigma_{j1}^{M(t)+1}, \sigma_{j2}^{M(t)+1}] = \left[ \beta \left( \sum_{j=1}^{n_u+n_o} (x_j - m_j^J)^2 \right)^{0.5}, \sigma_{j1}^{M(t)+1} + 0.1 \right] \tag{32}$$

where setting of $\sigma_{j1}^{M(t)+1}$ is similar to (29), and $\beta$ is also set to 0.5. Like (28), the initial value of $\sigma_{j2}^{M(t)+1}$ is assigned to generate a suitable FOU.

Previous studies [24], [25] have shown that different uncertain mean (STD) ranges generate different FOU areas and may influence the final results. In these studies, a constant uncertain range is manually selected in advance, and all MFs share the same range. In RSEIT2FNN, different MFs use different uncertain mean (STD) ranges, which are all automatically tuned using the following MF parameter learning algorithm.

### B. Parameter Learning

The parameter learning phase occurs concurrently with the structure learning phase. For each piece of incoming data, all the free RSEIT2FNN parameters are tuned, whether the rules are newly generated or originally existent. For clarity, consider only the $q$th network output. The objective of the parameter learning process is to minimize the error function

$$E = \frac{1}{2}[y_q'(t+1) - y_d(t+1)]^2. \tag{33}$$

Here, $y_q'(t+1)$ and $y_d(t+1)$ denote the RSEIT2FNN and desired outputs, respectively. The Karnik–Mendel iterative procedure for computing $y_{lq}'$ and $y_{rq}'$ has the premise that $\tilde{y}_{lq}^i$ and $\tilde{y}_{rq}^i$ have been rearranged in ascending order. During the parameter learning process, the consequent values $\tilde{y}_{lq}^i$ and $\tilde{y}_{rq}^i$ change, and their corresponding rule orders change accordingly. To update the parameters, it is necessary to know the precise locations of specific antecedent and consequent parameters, and this is very difficult to ascertain when the rule orders are different at each learning time step. The proposed rule-ordered Kalman filtering algorithm addresses this problem by keeping the original rule order during the parameter learning process. This is achieved by mapping the consequent values in ascending order with respect to the original rule order [see (17)]. According to this mapping, (18) and (21) are expressed by $\tilde{\boldsymbol{y}}_{lq}$ and $\tilde{\boldsymbol{y}}_{rq}$, i.e., with the consequent values arranged in the original rule order, despite their changes during the parameter learning process. Based on this original rule-ordered expression, the rule-ordered Kalman filtering algorithm is derived as follows. Equations (18) and (21) can be reexpressed as

$$y_{lq}' = \boldsymbol{\phi}_{lq}^{\boldsymbol{T}} \tilde{\boldsymbol{y}}_{lq}$$
$$\boldsymbol{\phi}_{lq} = \frac{\overline{\boldsymbol{\psi}}^T Q_l^T E_1^T E_1 Q_l + \underline{\boldsymbol{\psi}}^T Q_l^T E_2^T E_2 Q_l}{\boldsymbol{p}_l^T Q_l \overline{\boldsymbol{\psi}} + \boldsymbol{g}_l^T Q_l \underline{\boldsymbol{\psi}}} \in \Re^{M \times 1} \tag{34}$$

and

$$y_{rq}' = \boldsymbol{\phi}_{rq}^{\boldsymbol{T}} \tilde{\boldsymbol{y}}_{rq}$$
$$\boldsymbol{\phi}_{rq} = \frac{\underline{\boldsymbol{\psi}}^T Q_r^T E_3^T E_3 Q_r + \overline{\boldsymbol{\psi}}^T Q_r^T E_4^T E_4 Q_r}{\boldsymbol{p}_r^T Q_r \underline{\boldsymbol{\psi}} + \boldsymbol{g}_r^T Q_r \overline{\boldsymbol{\psi}}} \in \Re^{M \times 1} \tag{35}$$

respectively. Thus, the output $y_q'$ in (24) can be reexpressed as

$$\begin{aligned} y_q' &= \frac{1}{2}(y_{lq}' + y_{rq}') = \frac{1}{2}(\boldsymbol{\phi}_{lq}^{\boldsymbol{T}} \tilde{\boldsymbol{y}}_{lq} + \boldsymbol{\phi}_{rq}^{\boldsymbol{T}} \tilde{\boldsymbol{y}}_{rq}) \\ &= [\overline{\boldsymbol{\phi}}_{lq}^{\boldsymbol{T}} \overline{\boldsymbol{\phi}}_{rq}^{\boldsymbol{T}}] \begin{bmatrix} \tilde{\boldsymbol{y}}_{lq} \\ \tilde{\boldsymbol{y}}_{rq} \end{bmatrix} \\ &= [\overline{\phi}_{lq}^1 \cdots \overline{\phi}_{lq}^M \overline{\phi}_{rq}^1 \cdots \overline{\phi}_{rq}^M] \begin{bmatrix} \tilde{y}_{lq}^1 \\ \vdots \\ \tilde{y}_{lq}^M \\ \tilde{y}_{rq}^1 \\ \vdots \\ \tilde{y}_{rq}^M \end{bmatrix} \end{aligned} \tag{36}$$

where $\overline{\boldsymbol{\phi}}_{lq}^{\boldsymbol{T}} = 0.5\boldsymbol{\phi}_{lq}^{\boldsymbol{T}}$, and $\overline{\boldsymbol{\phi}}_{rq}^{\boldsymbol{T}} = 0.5\boldsymbol{\phi}_{rq}^{\boldsymbol{T}}$. According to (15) and (16), (36) can be further expressed as in (37), shown at the bottom of the page.

Since RSEIT2FNN rules are generated online, the dimension of $\tilde{\boldsymbol{y}}_{lq}$ in (37) increases with time, and the positions of $c_{jkq}^{i}$ and $s_{jkq}^{i}$ in the same vector change accordingly. To keep the positions of $c_{jkq}^{i}$ and $s_{jkq}^{i}$ constant in the vector, the vector components in (37) are rearranged in rule order in the proposed rule-ordered Kalman filtering algorithm. Let $\tilde{\boldsymbol{y}}_{\mathrm{TSK}} \in \Re^{2M(\Sigma_{j=0}^{n_u}(N_j+1)+\Sigma_{j=1}^{n_o}(O_j+1))\times 1}$ denote all of the consequent parameters $c_{jkq}^{i}$ and $s_{jkq}^{i}$, i.e.,

$$\begin{aligned}\tilde{\boldsymbol{y}}_{\mathrm{TSK}} = [&c_{00q}^{1}\cdots c_{(n_o+n_u)N_{n_o}q}^{1} s_{00q}^{1}\cdots s_{(n_o+n_u)N_{n_o}q}^{1}\cdots \\ &c_{00q}^{M}\cdots c_{(n_o+n_u)N_{n_o}q}^{M} s_{00q}^{M}\cdots s_{(n_o+n_u)N_{n_o}q}^{M}]^{T}\end{aligned} \tag{38}$$

where the parameters are positioned according to the rule order so that their positions remain constant as the rule numbers increase during the structure learning process. Equation (37) can then be expressed as

$$\begin{aligned}y_q' = &[\overline{\phi}_{c1}u_0\ldots\overline{\phi}_{c1}y_{pn_o}(t-O_{n_o}) - \overline{\phi}_{s1}|u_0|\ldots \\ &-\overline{\phi}_{s1}|y_{pn_o}(t-O_{n_o})|\ldots\ldots\overline{\phi}_{cM}u_0\ldots \\ &\overline{\phi}_{cM}y_{pn_o}(t-O_{n_o}) - \overline{\phi}_{sM}|u_0|\ldots \\ &-\overline{\phi}_{sM}|y_{pn_o}(t-O_{n_o})|]\tilde{\boldsymbol{y}}_{\mathrm{TSK}} \\ = &\overline{\boldsymbol{\phi}}'^{T}_{\boldsymbol{TSK}}\tilde{\boldsymbol{y}}_{\mathrm{TSK}}\end{aligned} \tag{39}$$

where $\overline{\phi}_{cq}^{j} = \overline{\phi}_{lq}^{j} + \overline{\phi}_{rq}^{j}$ and $\overline{\phi}_{sq}^{j} = \overline{\phi}_{rq}^{j} - \overline{\phi}_{lq}^{j}$, $j = 1,\ldots,M$. The consequent parameter vector $\tilde{\boldsymbol{y}}_{\mathrm{TSK}}$ is updated by executing the following rule-ordered Kalman filtering algorithm:

$$\begin{aligned}\tilde{\boldsymbol{y}}_{\mathrm{TSK}}(t+1) &= \tilde{\boldsymbol{y}}_{\mathrm{TSK}}(t) + S(t+1)\overline{\boldsymbol{\phi}}'_{\mathrm{TSK}}(t+1)(y^{d}(t+1) \\ &\quad - \overline{\boldsymbol{\phi}}'_{\mathrm{TSK}^{\mathrm{T}}}(t+1)\tilde{\boldsymbol{y}}_{\mathrm{TSK}}(t)) \\ S(t+1) &= \frac{1}{\lambda}\left[S(t) - \frac{S(t)\overline{\boldsymbol{\phi}}'_{\mathrm{TSK}}(t+1)\overline{\boldsymbol{\phi}}'^{\mathrm{T}}_{\mathrm{TSK}}(t+1)S(t)}{\lambda + \overline{\boldsymbol{\phi}}'^{\mathrm{T}}_{\mathrm{TSK}}(t+1)S(t)\overline{\boldsymbol{\phi}}'_{\mathrm{TSK}}}\right]\end{aligned} \tag{40}$$

where $0 < \lambda \leq 1$ is a forgetting factor ($\lambda = 0.9995$ in this paper). The dimensions of the vectors $\tilde{\boldsymbol{y}}_{\mathrm{TSK}}$ and $\overline{\boldsymbol{\phi}}'_{\mathrm{TSK}}$ and the matrix $S$ increase when a new rule evolves. When a new rule evolves, RSEIT2FNN augments $S(t)$ as in (41), shown at the bottom of this page, where $C$ is a large positive constant, and the size of the identity matrix $I$ is $2(\sum_{j=0}^{n_u}(N_j+1) + \sum_{j=1}^{n_o}(O_j+1)) \times 2(\sum_{j=0}^{n_u}(N_j+1) + \sum_{j=1}^{n_o}(O_j+1))$.

The RSEIT2FNN antecedent parameters are tuned by a gradient descent algorithm. Detailed learning equations can be found in the Appendix. Finally, it should be emphasized that after parameter update at each time step, rule consequent values $\tilde{\boldsymbol{y}}_{lq}$ and $\tilde{\boldsymbol{y}}_{rq}$ change. Therefore, the two permutation matrices $Q_l$ and $Q_r$ in (17) should change accordingly at each time step.

## IV. Simulations

This section describes four examples of RSEIT2FNN simulations. These examples include single-input–single-output (SISO) dynamic system identification (Example 1), chaotic series prediction (Example 2), MIMO dynamic system identification (Example 3), and real-time series prediction (Example 4). The performance of an RSEIT2FNN is compared with recurrent

$$y_q' = [\overline{\boldsymbol{\phi}}_{lq}^{\boldsymbol{T}} \quad \overline{\boldsymbol{\phi}}_{rq}^{\boldsymbol{T}}]\begin{bmatrix}\tilde{\boldsymbol{y}}_{lq}\\ \tilde{\boldsymbol{y}}_{rq}\end{bmatrix} = [\overline{\phi}_{lq}^{1}\cdots\overline{\phi}_{lq}^{M}\,\overline{\phi}_{rq}^{1}\cdots\overline{\phi}_{rq}^{M}]$$

$$\begin{bmatrix}
\sum_{j=0}^{n_u}\sum_{k=0}^{N_j} c_{jkq}^{1}u_j(t-k) + \sum_{j=1}^{n_o}\sum_{k=0}^{O_j} c_{(j+n_u)kq}^{1}y_{pj}(t-k) - \sum_{j=0}^{n_u}\sum_{k=0}^{N_j} s_{jkq}^{1}|u_j(t-k)| - \sum_{j=1}^{n_o}\sum_{k=0}^{O_j} s_{(j+n_u)kq}^{1}|y_{pj}(t-k)| \\
\vdots \\
\sum_{j=0}^{n_u}\sum_{k=0}^{N_j} c_{jkq}^{M}u_j(t-k) + \sum_{j=1}^{n_o}\sum_{k=0}^{O_j} c_{(j+n_u)kq}^{M}y_{pj}(t-k) - \sum_{j=0}^{n_u}\sum_{k=0}^{N_j} s_{jkq}^{M}|u_j(t-k)| - \sum_{j=1}^{n_o}\sum_{k=0}^{O_j} s_{(j+n_u)kq}^{M}|y_{pj}(t-k)| \\
\sum_{j=0}^{n_u}\sum_{k=0}^{N_j} c_{jkq}^{1}u_j(t-k) + \sum_{j=1}^{n_o}\sum_{k=0}^{O_j} c_{(j+n_u)kq}^{1}y_{pj}(t-k) + \sum_{j=0}^{n_u}\sum_{k=0}^{N_j} s_{jkq}^{1}|u_j(t-k)| + \sum_{j=1}^{n_o}\sum_{k=0}^{O_j} s_{(j+n_u)kq}^{1}|y_{pj}(t-k)| \\
\vdots \\
\sum_{j=0}^{n_u}\sum_{k=0}^{N_j} c_{jkq}^{M}u_j(t-k) + \sum_{j=1}^{n_o}\sum_{k=0}^{O_j} c_{(j+n_u)kq}^{M}y_{pj}(t-k) + \sum_{j=0}^{n_u}\sum_{k=0}^{N_j} s_{jkq}^{M}|u_j(t-k)| + \sum_{j=1}^{n_o}\sum_{k=0}^{O_j} s_{(j+n_u)kq}^{M}|y_{pj}(t-k)|
\end{bmatrix}. \tag{37}$$

$$\begin{aligned}S(t) &= \text{block diag}[S(t) \quad CI] \\ &\in \Re^{2(M+1)\left(\sum_{j=0}^{n_u}(N_j+1)+\sum_{j=1}^{n_o}(O_j+1)\right)\times 2(M+1)\left(\sum_{j=0}^{n_u}(N_j+1)+\sum_{j=1}^{n_o}(O_j+1)\right)}\end{aligned} \tag{41}$$

TABLE I
PERFORMANCE OF RSEIT2FNN AND OTHER FEEDFORWARD AND RECURRENT MODELS FOR SISO PLANT IDENTIFICATION IN EXAMPLE 1

| Models | Feedforward Type-1 FNN | Feedforward Type-2 FNN | ERNN [30] | RFNN [4] | WRFNN [5] | TRFN-S [7] | RSEIT2FNN -UD | RSEIT2FNN -UM |
|---|---|---|---|---|---|---|---|---|
| Rule number | 7 | 4 | - | 7 | 5 | 5 | 2 | 2 |
| Parameter number | 49 | 48 | 54 | 49 | 55 | 60 | 42 | 42 |
| Training RMSE | 0.007 | 0.0155 | 0.036 | 0.072 | 0.0574 | 0.0076 | 0.0033 | 0.0034 |
| Test RMSE | 0.032 | 0.038 | 0.078 | 0.128 | 0.083 | 0.029 | 0.0058 | 0.006 |

type-1 FNNs and feedforward type-1 and type-2 FNNs in these examples.

### A. *Example 1 (SISO Dynamic System Identification)*

This example uses the RSEIT2FNN to identify an SISO linear time-varying system, which is a problem that was introduced in [7]. The dynamic system with input delays is guided by the difference equation

$$y_{p1}(t+1) = 0.72y_{p1}(t) + 0.025y_{p1}(t-1)u_1(t-1) + 0.01u_1^2(t-2) + 0.2u_1(t-3). \quad (42)$$

This system has a single input ($n_u = 1$) and a single output ($n_o = 1$), and therefore, the two current variables $u_1(t)$ and $y_{p1}(t)$ are fed as inputs to the RSEIT2FNN input layer. The current output of the plant depends on three previous inputs and one previous output. Therefore, the lag numbers $N_1$ and $O_1$ in RSEIT2FNN consequent part are set to 3 and 1, respectively. The training procedure for the RSEIT2FNN uses the plant output $y_{p1}(t+1)$ as the desired output $y_d(t+1)$. In training the RSEIT2FNN, we use only ten epochs, each of which contains 900 time steps. As in [7], the input is an independently and identically distributed (i.i.d.) uniformly random sequence over $[-2, 2]$ for about half of the 900 time steps and a sinusoid given by $1.05\sin(\pi k/45)$ for the remaining time. There is no repetition in these 900 training data, i.e., we have different training sets for each epoch. This type of training is similar to an online training process, where there are a total number of 9000 online training time steps. There is no repeated training for the training dataset obtained in each time step. The structure learning threshold $f_{\text{th}}$ determines the number of fuzzy rules to be generated. For RSEIT2FNN-UM, two rules are generated when $f_{\text{th}}$ is set to 0.05. Table I shows the root-mean-squared error (RMSE) of the training data. To see the identification result, the following input used in [7] is also adopted for the test:

$$u_1(t) = \begin{cases} \sin\left(\dfrac{\pi t}{25}\right), & t < 250 \\ 1.0, & 250 \le t < 500 \\ -1.0, & 500 \le t < 750 \\ 0.3\sin\left(\dfrac{\pi t}{25}\right) + 0.1\sin\left(\dfrac{\pi t}{32}\right) & \\ \quad + 0.6\sin\left(\dfrac{\pi t}{10}\right), & 750 \le t < 1000. \end{cases} \quad (43)$$

Fig. 3 shows the outputs of the plant and the RSEIT2FNN-UM for these test inputs. Fig. 4 shows the test error $y_1'(t+1) - y_{p1}(t+1)$ between the outputs of the RSEIT2FNN-UM and the actual plant. Table I shows the network size, and training and test RMSEs of RSEIT2FNN-UM. Table I also shows performance of an RSEIT2FNN-UD with the same network size as the RSEIT2FNN-UM. The results show that these two networks have similar performance.

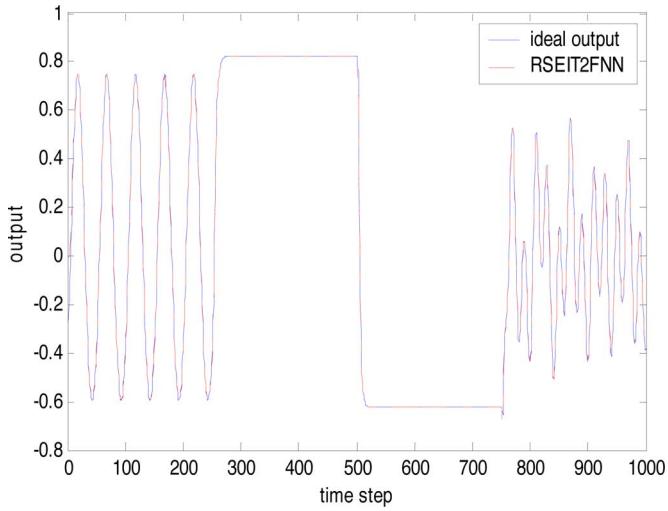



Fig. 3. Outputs of the dynamic plant (solid line) and RSEIT2FNN-UM (dotted line) in Example 1.

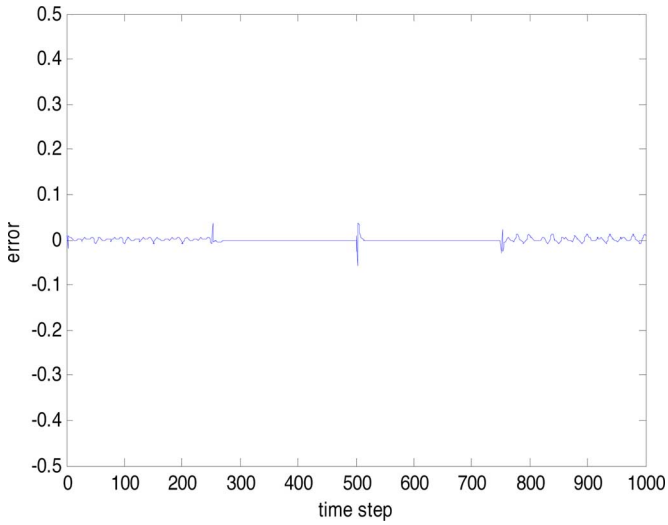



Fig. 4. Test errors between the RSEIT2FNN-UM and actual plant outputs.

The performance of RSEIT2FNN is compared with that of TSK-type feedforward type-1 and type-2 FNNs, a recurrent

TABLE II
INFLUENCE OF $f_{\rm th}$ ON THE PERFORMANCE OF AN SEIT2FNN-UM WITH $\beta = 0.5$

| $f_{th}$ | 0.05 | 0.15 | 0.25 | 0.3 |
|---|---|---|---|---|
| Rule number | 2 | 6 | 8 | 9 |
| Training RMSE | 0.0034 | 0.0018 | 0.0013 | 0.0011 |
| Test RMSE | 0.006 | 0.0054 | 0.0040 | 0.0046 |

TABLE III
INFLUENCE OF $\beta$ ON THE PERFORMANCE OF AN RSEIT2FNN-UM WITH $f_{\rm th} = 0.05$

| $\beta$ | 0.2 | 0.3 | 0.5 | 0.6 | 0.7 |
|---|---|---|---|---|---|
| Rule number | 10 | 4 | 2 | 2 | 2 |
| Training RMSE | 0.0013 | 0.0012 | 0.0034 | 0.0034 | 0.0035 |
| Test RMSE | 0.0112 | 0.0054 | 0.006 | 0.0075 | 0.0087 |

NN, and recurrent type-1 FNNs. The compared feedforward type-1 FNN is a self-constructing neural fuzzy inference network (SONFIN) [29], which is a powerful network with both structure and parameter learning. As in an RSEIT2FNN, the consequent part of a SONFIN is also trained using the Kalman filter algorithm. The feedforward type-2 FNN for comparison is the interval type-2 FNN [21], where all the network parameters are learned using the steepest descent algorithm. In the original interval type-2 FNN, the network structure is fixed and assigned in advance. Since an RSEIT2FNN uses structure learning for network design, the proposed structure learning in Section III-A is also incorporated in the interval type-2 FNN in this and the following examples. Comparisons are based on a similar network size, i.e., the total number of parameters in a feedforward type-1 FNN is similar to that in a feedforward interval type-2 FNN. The number of rule parameters in an interval type-2 FNN is larger than that in a type-1 FNN due to the use of additional free parameters in type-2 fuzzy sets and rule consequent part. Therefore, the total number of rules in a feedforward type-1 FNN is set to be larger than that in an interval type-2 FNN, as shown in Table I. The recurrent NN used for comparison is Elman's recurrent NN (ERNN) [30], which is applied to the same problem in [7]. The recurrent type-1 FNNs include the RFNN [4], the wavelet-based RFNN (WRFNN) [5], and the TSK-type recurrent fuzzy network with supervised learning (TRFN-S) [7]. All these networks use the same training data, test data, and number of training epochs as the RSEIT2FNN. Table I shows the number of rules, the total number of network parameters, and the training and test errors of these compared networks. Among all the recurrent type-1 networks being compared, the TRFN-S achieves the minimum test error. The results show that the RSEIT2FNN achieves smaller training and test errors than the other feedforward and recurrent networks.

We now analyze the practical computational cost of constructing an RSEIT2FNN. Since the functions in an RSEIT2FNN-UM are similar to those in an RSEIT2FNN-UD, only the former network is studied. All simulations are performed on an Intel 3.0 GHz dual CPU, and the programs are written in Visual C++. The total learning time of the RSEIT2FNN-UM mentioned before is 0.344 s, which is a very short time. We compare the computational costs for training the two representative networks SONFIN and TRFN-S for feedforward and recurrent type-1 FNNs. The SONFIN and TRFN-S take 0.891 and 4.407 s, respectively. The results show that the RSEIT2FNN-UM takes less training time than these two networks.

We consider the influence of the values of $f_{\rm th}$ and $\beta$ on the RSEIT2FNN-UM performance. The threshold $f_{\rm th}$ decides the number of rules in the RSEIT2FNN-UM. Table II shows the RSEIT2FNN-UM performance for different values of $f_{\rm th}$ when $\beta = 0.5$. Larger values of $f_{\rm th}$ generate larger numbers of rules and improve the learning performance of the network in general. However, when the value of $f_{\rm th}$ is too large, the performance saturates. One reason is that it is easier to get stuck in a local optimum when training a larger network. Another reason is that it requires a larger number of training epochs for network convergence. For a constant value of $f_{\rm th}$, smaller values of $\beta$ generate larger numbers of rules because of the smaller initial type-2 fuzzy set width. Table III shows the RSEIT2FNN-UM performance for different values of $\beta$ when $f_{\rm th} = 0.05$. The influence of rule number on network performance is similar to that discussed before. Table III shows that for a constant rule number of 2, the network performance is insensitive to variations in$\beta$. The general rule for selecting $f_{\rm th}$ and $\beta$ is to set a constant value of $\beta$ (e.g., $\beta = 0.5$ in this paper), first for the given problem and then to select $f_{\rm th}$ based on a compromise between network size and performance.

This example also studies the RSEIT2FNN test performance when the measured plant output $y_{p1}$ contains noise. The test also uses the control input sequence in (43). The added noise is artificially generated white Gaussian noise with three different STDs of 0.1, 0.5, and 0.7. There are 30 Monte Carlo realizations for statistical analysis. Table IV shows the statistical mean and STD (denoted as mean $\pm$ STD) of RSEIT2FNN-UM and RSEIT2FNN-UD for different noise levels. The results show that these two networks have similar performance. The primary MFs in both RSEIT2FNN-UM and RSEIT2FNN-UD are of Gaussian type. The numbers of additional flexible parameters provided by the FOU in each MF of both networks are identical and are tuned by the same parameter-learning algorithm. It is reasonable that these two networks have similar performance in Tables I and IV.

The performance of feedforward type-1 and type-2 FNNs and TRFN-S for the same noisy test patterns is compared with that of the RSEIT2FNN. Though the performance of different recurrent models is studied in Table I, this example only compares RSEIT2FNN with TRFN-S. The reason is that TRFN-S achieves the minimum test RMSE among the compared recurrent models in Table I. Table IV shows the average RMSEs of the feedforward and recurrent models for different noise levels. For the two feedforward FNNs, the test errors of the type-2 FNN are smaller than those of the type-1 FNN for different noise levels. The results show that the test errors of RSEIT2FNN are smaller than those of the other models tested for all of the test noise levels.

TABLE IV
PERFORMANCE OF RSEIT2FNN AND OTHER FEEDFORWARD AND RECURRENT MODELS WITH DIFFERENT NOISE LEVELS IN EXAMPLE 1

| Models | | Feedforward Type-1 FNN | Feedforward Type-2 FNN | TRFN-S | RSEIT2FNN -UD | RSEIT2FNN -UM |
|---|---|---|---|---|---|---|
| Rule number | | 7 | 4 | 5 | 2 | 2 |
| Parameter number | | 49 | 48 | 60 | 42 | 42 |
| Test RMSE | STD=0.1 | $0.145 \pm 0.002$ | $0.087 \pm 0.002$ | $0.097 \pm 0.002$ | $0.075 \pm 0.001$ | $0.072 \pm 0.001$ |
| | STD=0.3 | $0.309 \pm 0.008$ | $0.246 \pm 0.007$ | $0.276 \pm 0.006$ | $0.223 \pm 0.005$ | $0.217 \pm 0.007$ |
| | STD=0.5 | $0.501 \pm 0.011$ | $0.416 \pm 0.006$ | $0.453 \pm 0.012$ | $0.372 \pm 0.009$ | $0.358 \pm 0.009$ |

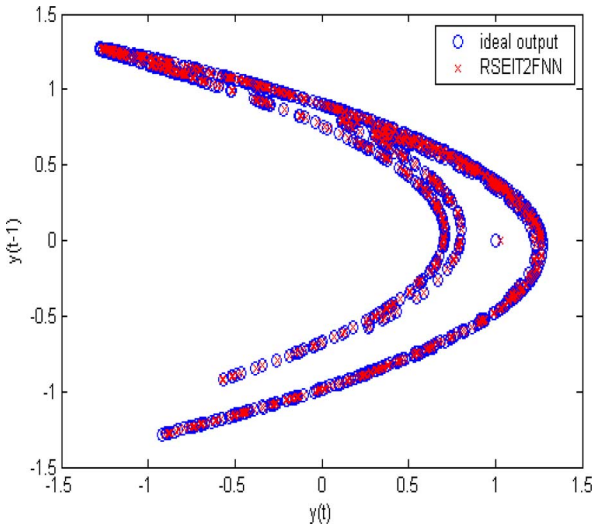



Fig. 5. Results of the phase plot for the chaotic system (◯) and RSEIT2FNN-UM (×).

### B. Example 2 (Chaotic Series Prediction)

The chaotic system is described by

$$y_{p1}(t+1) = -Py_{p1}^2(t) + Qy_{p1}(t-1) + 1.0. \tag{44}$$

The study [31] shows that the system produces a chaotic strange attractor when the parameters $P$ and $Q$ are set to 1.4 and 0.3, respectively. The system has no control input ($n_u = 0$) and a single output ($n_o = 1$); therefore, only the state $y_{p1}(t)$ is fed as input to the RSEIT2FNN input layer. The system is of second order with one delay, so the lag number $O_1$ in the RSEIT2FNN consequent part is set to one. The desired output is $y_d(t+1) = y_{p1}(t+1)$. Two thousand patterns are generated from the initial state $[y_{p1}(1), y_{p1}(0)] = [0.4, 0.4]$, where the first 1000 patterns are used for training, and the remaining 1000 patterns are used for testing. In RSEIT2FNN-UM training, the structure learning threshold $f_{\text{th}}$ is set to 0.3. After 90 epochs of training, six rules are generated. Fig. 5 shows the phase plane of the actual and RSEIT2FNN-UM prediction results for the test patterns. Table V shows the structure and training and test RMSEs of RSEIT2FNN-UM. The performance of an RSEIT2FNN-UD with the sane network size is also shown in Table V. Like the results in Example 1, Table V shows that RSEIT2FNN-UM and RSEIT2FNN-UD have similar performance. As in Example 1, the performance of the RSEIT2FNN is compared with that of feedforward type-1 and type-2 FNNs [21], [29] and recurrent type-1 FNNs, including RFNN [4], WRFNN [5], and TRFN-S [7]. These compared networks use the same number of training epochs and training and test data as in the RSEIT2FNN. Table V shows the numbers of rules and network parameters and training and test RMSEs of these compared networks. The results show that the RSEIT2FNN achieves better performance than other networks.

This example also studies the RSEIT2FNN test performance when the measured plant output $y_{p1}$ contains noise. The added noise is artificially generated white Gaussian noise with STD of 0.3, 0.5, and 0.7. Table VI shows the test RMSEs over 30 Monte Carlo realizations. The results in Table VI show that RSEIT2FNN-UM and RSEIT2FNN-UD have similar performance. For comparison, Table VI also shows the test RMSEs of feedforward type-1 and type-2 FNNs, and TRFN-S. The results show that the RMSE of the RSEIT2FNN is smaller than that of the other networks for all of the test noise levels.

### C. Example 3 (MIMO Dynamic System Identification)

The identified MIMO plant is the same as that used in [32]. The plant is described by

$$y_{p1}(t+1) = 0.5\left[\frac{y_{p1}(t)}{1+y_{p2}^2(t)} + u_1(t-1)\right]$$
$$y_{p2}(t+1) = 0.5\left[\frac{y_{p1}(t)y_{p2}(t)}{1+y_{p2}^2(t)} + u_2(t-1)\right]. \tag{45}$$

This plant has two inputs ($n_u = 2$) and two outputs ($n_o = 2$) so that the four current input and output variables $u_1(t)$, $u_2(t)$, $y_{p1}(t)$, and $y_{p2}(t)$ are fed as inputs to the RSEIT2FNN input layer. The current output of the plant depends on the control inputs with one time-step delay and current plant states. Therefore, the lag numbers $N_1$, $N_2$, $O_1$, and $O_2$ in the RSEIT2FNN consequent part are set to be 1, 1, 0, and 0, respectively. The two desired outputs for the RSEIT2FNN training are $y_{p1}(t+1)$ and $y_{p2}(t+1)$. During the training phase, the RSEIT2FNN is trained online from time step $t = 1$ to $t = 11\,000$. The two control inputs $u_1(t)$ and $u_2(t)$ are i.i.d. uniformly random sequences over $[-1.4, 1.4]$ from $t = 1$ to $t = 4000$ and sinusoid signals given by $\sin(\pi t/45)$ from $t = 4001$ to $t = 11\,000$. In the RSEIT2FNN-UM training, the structure learning threshold $f_{\text{th}}$ is set to 0.07. A larger network is generated when $f_{\text{th}}$ is larger than 0.1. The threshold value (0.07) is determined based on a compromise between network size and performance, as discussed in Example 1. After the training, three rules are generated. Table VII shows the structure and RMSE of

TABLE V
PERFORMANCE OF RSEIT2FNN AND OTHER FEEDFORWARD AND RECURRENT MODELS IN EXAMPLE 2

| Models | Feedforward Type-1 FNN | Feedforward Type-2 FNN | RFNN [4] | WRFNN [5] | TRFN-S [7] | RSEIT2FNN -UD | RSEIT2FNN -UM |
|---|---|---|---|---|---|---|---|
| Rule number | 15 | 8 | 15 | 7 | 6 | 6 | 6 |
| Parameter number | 60 | 56 | 60 | 70 | 66 | 60 | 60 |
| Training RMSE | 0.234 | 0.201 | 0.463 | 0.191 | 0.0296 | 0.0057 | 0.0043 |
| Test RMSE | 0.240 | 0.200 | 0.469 | 0.188 | 0.0245 | 0.0048 | 0.0047 |

TABLE VI
PERFORMANCE OF RSEIT2FNN AND OTHER FEEDFORWARD AND RECURRENT MODELS WITH DIFFERENT NOISE LEVELS IN EXAMPLE 2

| Models | | Feedforward Type-1 FNN | Feedforward Type-2 FNN | TRFN-S | RSEIT2FNN -UD | RSEIT2FNN -UM |
|---|---|---|---|---|---|---|
| Rule number | | 15 | 8 | 6 | 6 | 6 |
| Parameter number | | 60 | 56 | 66 | 60 | 60 |
| Test RMSE | STD=0.3 | $0.621 \pm 0.023$ | $0.617 \pm 0.013$ | $0.604 \pm 0.015$ | $0.528 \pm 0.016$ | $0.530 \pm 0.012$ |
| | STD=0.5 | $1.114 \pm 0.069$ | $0.917 \pm 0.022$ | $0.955 \pm 0.028$ | $0.808 \pm 0.021$ | $0.803 \pm 0.020$ |
| | STD=0.7 | $1.706 \pm 0.109$ | $1.180 \pm 0.024$ | $1.256 \pm 0.031$ | $1.026 \pm 0.031$ | $1.020 \pm 0.035$ |

TABLE VII
PERFORMANCE OF RSEIT2FNN AND OTHER RECURRENT MODELS FOR MIMO PLANT IDENTIFICATION IN EXAMPLE 3

| Models | Feedforward Type-1 FNN | Feedforward Type-2 FNN | MNN* [32] | RFNN [4] | WRFNN [5] | TRFN-S [7] | RSEIT2FNN -UM |
|---|---|---|---|---|---|---|---|
| Rule number | 7 | 4 | - | 13 | 7 | 7 | 3 |
| Parameter number | 126 | 128 | 131 | 182 | 182 | 189 | 126 |
| Training RMSE | 0.0422 | 0.0496 | -- | 0.0687 | 0.0449 | 0.0382 | 0.0036 |
| Test RMSE $y_{p1}$ | 0.0373 | 0.0411 | 0.0186 | 0.0824 | 0.0472 | 0.0396 | 0.0081 |
| Test RMSE $y_{p2}$ | 0.0480 | 0.0423 | 0.0327 | 0.0801 | 0.0502 | 0.0383 | 0.0113 |

* The total training time step number in a MNN is 77000.

RSEIT2FNN-UM. To test the identification result, the two control input sequences are as follows:

$$u_1(t) = u_2(t) = \begin{cases} \sin\left(\dfrac{\pi t}{25}\right), & 1001 \le t < 1250 \\ 1.0, & 1250 \le t < 1500 \\ -1.0, & 1500 \le t < 1750 \\ 0.3\sin\left(\dfrac{\pi t}{25}\right) + 0.1\sin\left(\dfrac{\pi t}{32}\right) & \\ \quad + 0.6\sin\left(\dfrac{\pi t}{10}\right), & 1750 \le t < 2000. \end{cases} \tag{46}$$

Fig. 6 shows the test results. Table VII shows the test RMSEs for outputs $y_{p1}$ and $y_{p2}$.

For comparison, Table VII shows the performance of the memory NN (MNN) [32], feedforward type-1 and type-2 FNNs, and recurrent type-1 FNNs studied in Example 2. The MNN is a kind of recurrent NN, and has been applied to the same problem in [32]. These networks use a total number of 11 000 training time steps as in the RSEIT2FNN-UM, except in the case of the MNN, where a total number of 77 000 time steps is used for training in [32]. The results in Table VII show that the RSEIT2FNN-UM achieves smaller RMSEs for both outputs than these feedforward and recurrent networks.

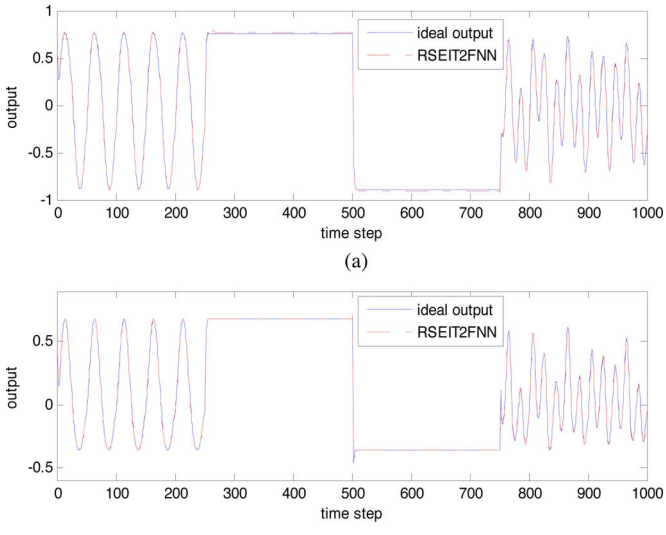



Fig. 6. Outputs of the MIMO plant (solid curve) and RSEIT2FNN-UM (dotted curve) in Example 3. (a) Output $y_{p1}$. (b) Output $y_{p2}$.

This example also studies the RSEIT2FNN-UM test performance when the measured plant outputs $y_{p1}$ and $y_{p2}$ contain noise. The added noise is artificially generated white Gaussian noise with STD of 0.3, 0.5, and 0.7. Table VIII shows the average test RMSEs of the RSEIT2FNN-UM, feedforward type-1 and type-2 FNNs, and TRFN-S over 30 Monte Carlo realizations. The results show that the RMSEs of RSEIT2FNN-UM

TABLE VIII
PERFORMANCE OF RSEIT2FNN AND TRFN-S WITH DIFFERENT NOISE LEVELS IN EXAMPLE 3

| Models | | Feedforward Type-1 FNN | Feedforward Type-2 FNN | TRFN-S | RSEIT2FNN -UM |
|---|---|---|---|---|---|
| Rule number | | 13 | 3 | 7 | 3 |
| Parameter number | | 182 | 120 | 189 | 126 |
| Test RMSE $y_{p1}$ | STD=0.3 | $0.256 \pm 0.005$ | $0.189 \pm 0.003$ | $0.188 \pm 0.007$ | $0.160 \pm 0.003$ |
| | STD=0.5 | $0.417 \pm 0.010$ | $0.307 \pm 0.006$ | $0.316 \pm 0.010$ | $0.262 \pm 0.005$ |
| | STD=0.7 | $0.578 \pm 0.013$ | $0.420 \pm 0.009$ | $0.427 \pm 0.014$ | $0.368 \pm 0.006$ |
| Test RMSE $y_{p2}$ | STD=0.3 | $0.197 \pm 0.003$ | $0.145 \pm 0.003$ | $0.143 \pm 0.005$ | $0.090 \pm 0.002$ |
| | STD=0.5 | $0.322 \pm 0.006$ | $0.233 \pm 0.004$ | $0.226 \pm 0.006$ | $0.155 \pm 0.004$ |
| | STD=0.7 | $0.443 \pm 0.011$ | $0.323 \pm 0.006$ | $0.298 \pm 0.008$ | $0.227 \pm 0.008$ |

TABLE IX
PERFORMANCE OF RSEIT2FNN AND DIFFERENT MODELS FOR THE SERIES-E PREDICTION PROBLEM IN EXAMPLE 4

| Models | ES | Neural Network | PMRS [34] | Feedforward Type-1 FNN | Feedforward Type-2 FNN | RSEIT2FNN -UM |
|---|---|---|---|---|---|---|
| Test RMSE | 0.182 | 0.279 | 0.123 | 0.064 | 0.069 | 0.056 |

are smaller than those of the comparable networks when there is noise.

### D. Example 4 (Practical Time Series Prediction)

This example studies the performance of an RSEIT2FNN-UM for a real-world series database. The Series-E from the Sante Fe Time Series [33] is used (database Web site: http://www-psych.standford.edu/~andress/Time-Series/). This series is a set of astrophysical data (variation in light intensity of a star). The objective is to predict the intensity of the star at time $t+1$, $y_{p1}(t+1)$, according to its past intensities. This benchmark series is selected because it is very noisy and discontinuous. According to [34], 2048 observations were collected, of which 90% were used for training and the remaining 10% for testing.

Since the appropriate number of lagged intensities for prediction is unknown in advance for this practical series, the lag number $O_1$ in the RSEIT2FNN-UM is simply set to zero. Only the intensity $y_{p1}(t)$ is fed as input to the RSEIT2FNN-UM input layer, because the system has no external input ($n_u = 0$), i.e., only $y_{p1}(t)$ is fed as input to the RSEIT2FNN-UM for predicting $y_{p1}(t+1)$, and the past values $y_{p1}(t-j)$ are automatically memorized in the feedback loops. If the appropriate lag number $O_1$ is known in advance, then more past values other than $y_{p1}(t)$ can be included in the RSEIT2FNN-UM consequent part for a better prediction performance. The threshold $f_{\text{th}}$ is set to 0.03, and the number of rules is 6 (48 parameters in total) after 100 epochs of training. Table IX shows the test RMSE of the RSEIT2FNN-UM. Fig. 7 shows the actual and RSEIT2FNN-UM-predicted intensities. For comparison, Table IX also shows the test RMSEs of the feedforward type-1 and type-2 FNNs using the same input–output pairs. The numbers of rules (parameters) in the feedforward type-1 and type-2 FNNs are 12 (48) and 7 (49), respectively. The test error of the RSEIT2FNN-UM is smaller than those of these two networks.

A model called pattern modeling and recognition system (PMRS) was proposed in [34] to predict the same series. The prediction results using feedforward NN and statistical exponential smoothing (ES) method are also reported in that study. In these methods, an appropriate number of past intensities should be determined for each model input, which burdens model design effort. For example, the NN uses the past five intensities as network inputs. Table IX shows the test RMSEs of these three models, all of which have larger errors than the RSEIT2FNN-UM and the two feedforward FNNs.

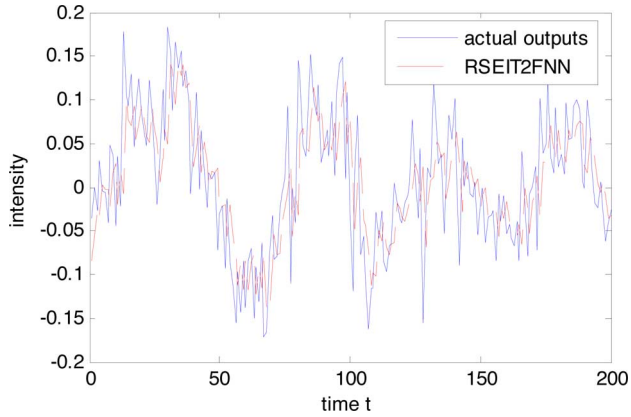



Fig. 7. Prediction results of the RSEIT2FNN-UM for the Series-E problem in Example 4.

## V. CONCLUSION

This paper proposes a new recurrent type-2 FNN, i.e., the RSEIT2FNN. In contrast to existing feedforward type-2 FNNs, this network is especially useful for handling problems with temporal properties. For RSEIT2FNN learning, there is no need to determine the RSEIT2FNN structure in advance because the proposed structure learning ability enables the RSEIT2FNN to evolve its structure online. Moreover, the proposed rule-ordered Kalman filter algorithm helps tune the consequent parameters online and improves learning accuracy. Simulation results show that the RSEIT2FNN achieves a better performance than existing recurrent type-1 FNNs in both noise-free and noisy environments. The FOU in the RSEIT2FNN helps handle the numerical uncertainty associated with system inputs and outputs.

Therefore, the RSEIT2FNN has the potential to achieve better performance than type-1 fuzzy systems when dealing with noisy data, as demonstrated in the examples given in Section IV. Future studies will theoretically analyze the learning convergence of the RSEIT2FNN and examine possible practical applications of the RSEIT2FNN to temporal problems with noise or uncertainty.

## APPENDIX

This Appendix derives the antecedent parameter learning equations using a gradient descent algorithm. For convenience in notation of the gradient descent results, (18) can be reexpressed according to [21] as

$$y'_{lq} = \frac{\overline{\boldsymbol{\psi}}^T \boldsymbol{a}_{lq} + \underline{\boldsymbol{\psi}}^T \boldsymbol{b}_{lq}}{\overline{\boldsymbol{\psi}}^T \boldsymbol{c}_{lq} + \underline{\boldsymbol{\psi}}^T \boldsymbol{d}_{lq}} \tag{A1}$$

where

$$\boldsymbol{a}_{lq} = Q_l^T E_1^T E_1 Q_l \tilde{\boldsymbol{y}}_{lq} \in \Re^{M\times 1}$$
$$\boldsymbol{b}_{lq} = Q_l^T E_2^T E_2 Q_l \tilde{\boldsymbol{y}}_{lq} \in \Re^{M\times 1} \tag{A2}$$
$$\boldsymbol{c}_{lq} = Q_l^T \boldsymbol{p}_l \in \Re^{M\times 1} \qquad \boldsymbol{d}_{lq} = Q_l^T \boldsymbol{g}_l \in \Re^{M\times 1}. \tag{A3}$$

Similarly, (21) can be reexpressed as

$$y'_{rq} = \frac{\underline{\boldsymbol{\psi}}^T \boldsymbol{a}_{rq} + \overline{\boldsymbol{\psi}}^T \boldsymbol{b}_{rq}}{\underline{\boldsymbol{\psi}}^T \boldsymbol{c}_{rq} + \overline{\boldsymbol{\psi}}^T \boldsymbol{d}_{rq}} \tag{A4}$$

where

$$\boldsymbol{a}_{rq} = Q_r^T E_3^T E_3 Q_r \tilde{\boldsymbol{y}}_{rq} \in \Re^{M\times 1}$$
$$\boldsymbol{b}_{rq} = Q_r^T E_4^T E_4 Q_r \tilde{\boldsymbol{y}}_{rq} \in \Re^{M\times 1} \tag{A5}$$
$$\boldsymbol{c}_{rq} = Q_r^T \boldsymbol{p}_r \in \Re^{M\times 1}, \qquad \boldsymbol{d}_{rq} = Q_r^T \boldsymbol{g}_r \in \Re^{M\times 1}. \tag{A6}$$

Using the gradient descent algorithm, we have

$$\lambda_q^i(t+1) = \lambda_q^i(t) - \eta \frac{\partial E}{\partial \lambda_q^i(t)} \tag{A7}$$

where $\eta$ is a learning constant ($\eta = 0.08$ in this paper), and

$$\frac{\partial E}{\partial \lambda_q^i} = \frac{\partial E}{\partial y'_q}\left(\frac{\partial y'_q}{\partial y'_{lq}}\frac{\partial y'_{lq}}{\partial \lambda_q^i} + \frac{\partial y'_q}{\partial y'_{rq}}\frac{\partial y'_{rq}}{\partial \lambda_q^i}\right)$$
$$= \frac{1}{2}(y'_q - y_d)\left[\left(\frac{\partial y'_{lq}}{\partial \overline{\psi}_q^i} + \frac{\partial y'_{rq}}{\partial \overline{\psi}_q^i}\right)\frac{\partial \overline{\psi}_q^i}{\partial \lambda_q^i} + \left(\frac{\partial y'_{lq}}{\partial \underline{\psi}_q^i} + \frac{\partial y'_{rq}}{\partial \underline{\psi}_q^i}\right)\frac{\partial \underline{\psi}_q^i}{\partial \lambda_q^i}\right] \tag{A8}$$

where

$$\frac{\partial y'_{lq}}{\partial \overline{\psi}_q^i} = \frac{a_{lqi} - y'_{lq} c_{lqi}}{\overline{\boldsymbol{\psi}}^T \boldsymbol{c}_{lq} + \underline{\boldsymbol{\psi}}^T \boldsymbol{d}_{lq}} \quad \frac{\partial y'_{rq}}{\partial \overline{\psi}_q^i} = \frac{b_{rqi} - y'_{rq} d_{rqi}}{\underline{\boldsymbol{\psi}}^T \boldsymbol{c}_{rq} + \overline{\boldsymbol{\psi}}^T \boldsymbol{d}_{rq}} \tag{A9}$$

$$\frac{\partial y'_{lq}}{\partial \underline{\psi}_q^i} = \frac{b_{lqi} - y'_{lq} d_{lqi}}{\overline{\boldsymbol{\psi}}^T \boldsymbol{c}_{lq} + \underline{\boldsymbol{\psi}}^T \boldsymbol{d}_{lq}} \quad \frac{\partial y'_{rq}}{\partial \underline{\psi}_q^i} = \frac{a_{rqi} - y'_{rq} c_{rqi}}{\underline{\boldsymbol{\psi}}^T \boldsymbol{c}_{rq} + \overline{\boldsymbol{\psi}}^T \boldsymbol{d}_{rq}} \tag{A10}$$

$$\frac{\partial \overline{\psi}_q^i}{\partial \lambda_q^i} = \overline{f}^i \quad \frac{\partial \underline{\psi}_q^i}{\partial \lambda_q^i} = \underline{f}^i. \tag{A11}$$

Let $w_j^i$ denote a parameter in the $i$th interval type-2 fuzzy set $\tilde{A}_j^i$ in input variable $x_j$. This parameter is updated as follows:

$$w_j^i(t+1) = w_j^i(t) - \eta \frac{\partial E}{\partial w_j^i(t)} \tag{A12}$$

where

$$\frac{\partial E}{\partial w_j^i(t)} = \frac{1}{2}(y'_q - y_d)\left[\left(\frac{\partial y'_{lq}}{\partial \overline{\psi}_q^i} + \frac{\partial y'_{rq}}{\partial \overline{\psi}_q^i}\right)\frac{\partial \overline{\psi}_q^i}{\partial w_j^i} + \left(\frac{\partial y'_{lq}}{\partial \underline{\psi}_q^i} + \frac{\partial y'_{rq}}{\partial \underline{\psi}_q^i}\right)\frac{\partial \underline{\psi}_q^i}{\partial w_j^i}\right]. \tag{A13}$$

1) *RSEIT2FNN with uncertain mean (RSEIT2FNN-UM):* The parameters $m_{j1}^i$, $m_{j2}^i$, and $\sigma_j^i$ in (2) and (3) are updated according to (A12) and (A13). If $w_j^i = m_{j1}^i$, then we have

$$\frac{\partial \overline{\psi}_q^i}{\partial w_j^i} = \frac{\partial \overline{\psi}_q^i}{\partial m_{j1}^i} = \frac{\partial \overline{\psi}_q^i}{\partial \overline{f}^i}\frac{\partial \overline{f}^i}{\partial \overline{\mu}_j^i}\frac{\partial \overline{\mu}_j^i}{\partial m_{j1}^i}$$
$$= \begin{cases} \lambda_q^i \overline{f}^i \times \dfrac{x_j - m_{j1}^i}{(\sigma_j^i)^2}, & x_j \le m_{j1}^i \\ 0, & \text{otherwise} \end{cases} \tag{A14}$$

$$\frac{\partial \underline{\psi}_q^i}{\partial w_j^i} = \frac{\partial \underline{\psi}_q^i}{\partial m_{j1}^i} = \frac{\partial \underline{\psi}_q^i}{\partial \underline{f}^i}\frac{\partial \underline{f}^i}{\partial \underline{\mu}_j^i}\frac{\partial \underline{\mu}_j^i}{\partial m_{j1}^i}$$
$$= \begin{cases} \lambda_q^i \underline{f}^i \times \dfrac{x_j - m_{j1}^i}{(\sigma_j^i)^2}, & x_j > \dfrac{m_{j1}^i + m_{j2}^i}{2} \\ 0, & \text{otherwise.} \end{cases} \tag{A15}$$

Similarly, if $w_j^i = m_{j2}^i$, then we have

$$\frac{\partial \overline{\psi}_q^i}{\partial w_j^i} = \frac{\partial \overline{\psi}_q^i}{\partial m_{j2}^i} = \begin{cases} \lambda_q^i \overline{f}^i \times \dfrac{x_j - m_{j2}^i}{(\sigma_j^i)^2}, & x_j > m_{j2}^i \\ 0, & \text{otherwise} \end{cases} \tag{A16}$$

$$\frac{\partial \underline{\psi}_q^i}{\partial m_{j2}^i} = \begin{cases} \lambda_q^i \underline{f}^i \times \dfrac{x_j - m_{j2}^i}{(\sigma_j^i)^2}, & x_j \le \dfrac{m_{j1}^i + m_{j2}^i}{2} \\ 0, & \text{otherwise.} \end{cases} \tag{A17}$$

If $w_j^i = \sigma_j^i$, then we have

$$\frac{\partial \overline{\psi}_q^i}{\partial w_j^i} = \frac{\partial \overline{\psi}_q^i}{\partial \sigma_j^i} = \begin{cases} \lambda_q^i \overline{f}^i \times \dfrac{(x_j - m_{j1}^i)^2}{(\sigma_j^i)^3}, & x_j < m_{j1}^i \\ \lambda_q^i \overline{f}^i \times \dfrac{(x_j - m_{j2}^i)^2}{(\sigma_j^i)^3}, & x_j > m_{j2}^i \\ 0, & \text{otherwise} \end{cases} \tag{A18}$$

and

$$\frac{\partial \underline{\psi}_q^i}{\partial w_j^i} = \frac{\partial \underline{\psi}_q^i}{\partial \sigma_j^i} = \begin{cases} \lambda_q^i \underline{f}^i \times \dfrac{(x_j - m_{j2}^i)^2}{(\sigma_j^i)^3}, & x_j \le \dfrac{m_{j1}^i + m_{j2}^i}{2} \\ \lambda_q^i \underline{f}^i \times \dfrac{(x_j - m_{j1}^i)^2}{(\sigma_j^i)^3}, & x_j > \dfrac{m_{j1}^i + m_{j2}^i}{2}. \end{cases} \tag{A19}$$

2) *RSEIT2FNN with uncertain STD (RSEIT2FNN-UD):* The parameters $m_j^i$, $\sigma_{j1}^i$, and $\sigma_{j2}^i$ in (4) and (5) are updated according to (A12) and (A13). If $w_j^i = m_j^i$, then we have

$$\frac{\partial \overline{\psi}_q^i}{\partial m_j^i} = \lambda_q^i \overline{f}^i \times \frac{x_j - m_j^i}{(\sigma_{j2}^i)^2} \quad \text{and} \quad \frac{\partial \underline{\psi}_q^i}{\partial m_j^i} = \lambda_q^i \underline{f}^i \times \frac{x_j - m_j^i}{(\sigma_{j1}^i)^2}. \tag{A20}$$

If $w_j^i = \sigma_{j1}^i$, then we have

$$\frac{\partial \overline{\psi}_q^i}{\partial \sigma_{j1}^i} = 0 \quad \text{and} \quad \frac{\partial \underline{\psi}_q^i}{\partial \sigma_{j1}^i} = \lambda_q^i \underline{f}^i \times \frac{(x_j - m_j^i)^2}{(\sigma_{j1}^i)^3}. \tag{A21}$$

If $w_j^i = \sigma_{j2}^i$, then we have

$$\frac{\partial \overline{\psi}_q^i}{\partial \sigma_{j2}^i} = \lambda_q^i \overline{f}^i \times \frac{(x_j - m_j^i)^2}{(\sigma_{j2}^i)^3} \quad \text{and} \quad \frac{\partial \underline{\psi}_q^i}{\partial \sigma_{j2}^i} = 0. \tag{A22}$$

## REFERENCES


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

[2] 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, Feb. 1999.

[3] 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. 3, pp. 1348–1359, Jun. 2004.

[4] 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.

[5] 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.

[6] 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.

[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] 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, 2006.

[9] 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.

[10] N. N. Karnik, J. M. Mendel, and Q. Liang, "Type-2 fuzzy logic systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 6, pp. 643–658, Dec. 1999.

[11] J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic System: Introduction and New Directions*. Upper Saddle River, NJ: Prentice–Hall, 2001.

[12] J. M. Mendel and R. I. John, "Type-2 fuzzy sets made simple," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 117–127, Apr. 2002.

[13] J. M. Mendel, "Type-2 fuzzy sets and systems: An overview," *IEEE Comput. Intell. Mag.*, vol. 2, no. 1, pp. 20–29, Feb. 2007.

[14] R. John and S. Coupland, "Type-2 fuzzy logic: A historical view," *IEEE Comput. Intell. Mag.*, vol. 2, no. 1, pp. 57–62, Feb. 2007.

[15] Q. Liang and J. M. Mendel, "Equalization of nonlinear time-varying channels using type-2 fuzzy adaptive filters," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 5, pp. 551–563, Oct. 2000.

[16] H. Hagras, "A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 4, pp. 524–539, Aug. 2004.

[17] J. Zeng and Z. Q. Liu, "Type-2 fuzzy hidden Markov models and their application to speech recognition," *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 3, pp. 454–467, Jun. 2006.

[18] C. Hwang and F. C.-H. Rhee, "Uncertain fuzzy clustering: Interval type-2 fuzzy approach to C-means," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 1, pp. 107–120, Feb. 2007.

[19] C. H. Lee, Y. C. Lin, and W. Y. Lai, "Systems identification using type-2 fuzzy neural network (type-2 FNN) systems," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom.*, 2003, vol. 3, pp. 1264–1269.

[20] C. H. Wang, C. S. Cheng, and T. T. Lee, "Dynamical optimal training for interval type-2 fuzzy neural network (T2FNN)," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 3, pp. 1462–1477, Jun. 2004.

[21] J. M. Mendel, "Computing derivatives in interval type-2 fuzzy logic system," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 1, pp. 84–98, Feb. 2004.

[22] H. Hagras, "Comments on dynamical optimal training for interval type-2 fuzzy neural network (T2FNN)," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 5, pp. 1206–1209, Oct. 2006.

[23] G. M. Mendez and O. Castillo, "Interval type-2 TSK fuzzy logic systems using hybrid learning algorithm," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, May 22–25, 2005, pp. 230–235.

[24] J. Zeng and Z. Q. Liu, "Type-2 fuzzy Markov random fields and their application to handwritten Chinese character recognition," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 3, pp. 747–760, Jun. 2008.

[25] J. Zeng, L. Xie, and Z. Q. Liu, "Type-2 fuzzy Gaussian mixture models," *Pattern Recognit.*, vol. 41, no. 12, pp. 3636–3643, 2008.

[26] Q. Liang and J. M. Mendel, "Interval type-2 fuzzy logic systems: Theory and design," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 5, pp. 535–550, Oct. 2000.

[27] J. M. Mendel, R. I. John, and F. Liu, "Interval type-2 fuzzy logic systems made simple," *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 6, pp. 808–821, Dec. 2006.

[28] C. F. Juang, S. H. Chiu, and S. W. Chang, "A self-organizing TS-type fuzzy network with support vector learning and its application to classification problems," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 5, pp. 998–1008, Oct. 2007.

[29] 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. 12–32, Feb. 1998.

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

[31] G. Chen, Y. Chen, and H. Ogmen, "Identifying chaotic systems via a Wiener-type cascade model," *IEEE Trans. Control Syst. Mag.*, vol. 17, no. 5, pp. 29–36, Oct. 1997.

[32] P. S. Sastry, G. Ssntharam, and K. P. Unnikrishnan, "Memory neural networks for identification and control of dynamic systems," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 306–319, Mar. 1994.

[33] A. S. Weigend and N. A. Gersehnfield, *Time Series Prediction: Forecasting the Future and Understanding the Past*. Reading, MA: Addison-Wesley, 1994.

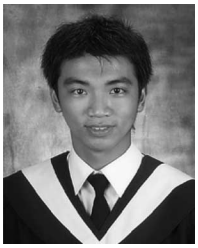
[34] S. Singh, "Noise impact on time-series forecasting using an intelligent pattern matching technique," *Pattern Recognit.*, vol. 32, no. 8, pp. 1389–1398, Aug. 1999.



**Chia-Feng Juang** (M'00–SM'08) received the B.S. and Ph.D. degrees in control engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in 1993 and 1997, respectively.

Since 2001, he has been with the Department of Electrical Engineering, National Chung-Hsing University (NCHU), Taichung, Taiwan, where he has been a Professor since 2007. He has authored or coauthored three book chapters, more than 55 refereed journal papers, and 55 conference papers. He has been a Referee for more than 45 international journals. He is currently a member of the Editorial Advisory Boards of the *Open Cybernetics and Systemics Journal* and the *Open Automation and Control Journal*. He is also a member of the Editorial Boards of the *International Journal of Computational Intelligence in Control* and the *Journal of Advanced Research in Evolutionary Algorithms*. He is an Area Editor of the *International Journal of Intelligent Systems Science and Technology*. His current research interests include computational intelligence (CI), intelligent control, computer vision, speech signal processing, and implementation of CI techniques using field-programmable gate arrays chips.

Dr. Juang was the recipient of the Youth Automatic Control Engineering Award from the Chinese Automatic Control Society, Taiwan, in 2006 and the Outstanding Youth Teacher Award from the NCHU in 2007. Six of his published journal papers were recognized as highly cited papers according to the Information Sciences Institute database in 2008.



**Ren-Bo Huang** received the B.S. degree in electrical engineering from the National United University, Miaoli, Taiwan, in 2007. He is currently working toward the M.S. degree in electrical engineering with the National Chung-Hsing University, Taichung, Taiwan.

His current research interests include type-2 neural fuzzy systems and field-programmable gate array chip design.

**Yang-Yin Lin** received the M.S. degree in electrical engineering from the National Chung-Hsing University, Taichung, Taiwan, in 2008. He is currently working toward the Ph.D. degree with the Department of Electrical Engineering, National Chiao-Tung University, Hsinchu, Taiwan.

His current research interests include neural networks and fuzzy systems.