A Mutually Recurrent Interval Type-2 Neural Fuzzy System (MRIT2NFS) With Self-Evolving Structure and Parameters

Yang-Yin Lin, Jyh-Yeong Chang, *Member, IEEE*, Nikhil R. Pal, *Fellow, IEEE*, and Chin-Teng Lin, *Fellow, IEEE*

***Abstract*—In this paper, a mutually recurrent interval type-2 neural fuzzy system (MRIT2NFS) is proposed for the identification of nonlinear and time-varying systems. The MRIT2NFS uses type-2 fuzzy sets in order to enhance noise tolerance of the system. In the MRIT2NFS, the antecedent part of each recurrent fuzzy rule is defined using interval type-2 fuzzy sets, and the consequent part is of the Takagi–Sugeno–Kang type with interval weights. The antecedent part of MRIT2NFS forms a local internal feedback and interaction loop by feeding the rule firing strength of each rule to others including itself. The consequent is a linear combination of exogenous input variables. The learning of MRIT2NFS starts with an empty rule base and all rules are learned online via structure and parameter learning. The structure learning of MRIT2NFS uses online type-2 fuzzy clustering. For parameter learning, the consequent part parameters are tuned by rule-ordered Kalman filter algorithm to reinforce parameter learning ability. The type-2 fuzzy sets in the antecedent and weights representing the mutual feedback are learned by the gradient descent algorithm. After the training, a weight-elimination scheme eliminates feedback connections that do not have much effect on the network behavior. This method can efficiently remove redundant recurrence and interaction weights. Finally, the MRIT2NFS is used for system identification under both noise-free and noisy environments. For this, we consider both time series prediction and nonlinear plant modeling. Compared with type-1 recurrent fuzzy neural networks, simulation results show that our approach produces smaller root-mean-squared errors using the same number of iterations.**

***Index Terms*—Dynamic system identification, on-line fuzzy clustering, recurrent neural fuzzy systems, type-2 fuzzy systems.**

I. Introduction

THERE are many real-life problems that are dynamic systems, and hence, identification of nonlinear dynamic systems is a very important research problem. In the control area, we usually encounter the problem of identification and control of dynamic systems. Since for a dynamic system, the control output is a combination of past inputs and/or outputs, identification and modeling of such a system is not as straightforward as that for a static/algebraic system. To deal with the temporal characteristics of dynamic systems, some recurrent fuzzy neural networks (RFNNs) have already been proposed [1]–[12] and have been shown to outperform feedforward FNNs and recurrent neural networks. Recently, a considerable research effort is being devoted toward developing recurrent neural-fuzzy models that are separated into two major categories. One category of recurrent FNNs uses feedback from the network output as the recurrence structure [2]–[4]. The other approach of recurrent FNNs uses feedback from internal state variables as its recurrence structure [5]–[12]. In [2], a recurrent neural-fuzzy system is proposed, where the consequent of a rule is a linear model in an autoregressive form with exogenous inputs. In [3] and [4], the authors have proposed an output-recurrent fuzzy neural network where the output values are fed back as input values. In [5] and [6], the recurrence property is achieved by feeding the output of each membership function (MF) back to itself; therefore, each membership value is dependent on its past values. In [8], a dynamic fuzzy neural network is proposed, where consequent parts are recurrent neural networks with local internal feedback. Recurrent self-organizing neural fuzzy inference network [7] and Takagi–Sugeno–Kang (TSK)-type recurrent fuzzy network (TRFN) [9], [10] use a global feedback structure, where the firing strengths of all rules are summed and fed back as internal network inputs. The recurrent self-evolving fuzzy neural network with local feedbacks (RSEFNN-LF) is proposed in [11] for dynamic system identification, where the recurrent firing values are influenced by both prior and current values.

All of the recurrent FNNs that we have discussed so far use type-1 fuzzy sets. In recent years, studies on type-2 fuzzy logic systems (FLS) have drawn much attention [13]–[17], [48]–[57]. Type-2 FLSs are extensions of type-1 FLSs, where the MFs involved in the fuzzy rules are type-2 fuzzy sets. We shall refer to such rules as type-2 fuzzy rules or type-2 rules. The membership values of a type-2 fuzzy set are type-1 fuzzy sets. Type-2 FLSs appear to be more promising than their type-1 counterparts in handling uncertainties, which allow researchers to model and minimize the effect of uncertainties associated with the rule-base system, and have already been successfully applied in several areas [17]–[21], [45]–[47], [58]–[61]. Usually, the T2FNN is computationally more expensive than that of its type-1 counterpart primarily due to the complexity of type reduction from type-2 to type-1. An interval type-2 fuzzy set (IT2FS) is a special case

Manuscript received September 5, 2012; revised January 7, 2013 and March 11, 2013; accepted March 14, 2013. Date of publication April 1, 2013; date of current version May 29, 2013. This work of N. R. Pal was supported by Grants 102W963 and NSC-101-2911-I-009-101 of the Brain Research Center, National Chiao Tung University, Hsinchu, Taiwan. Y.-Y. Lin, J.-Y. Chang, and C.-T. Lin contributed equally to this work.

Y.-Y. Lin, J.-Y. Chang, and C.-T. Lin 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).

N. R. Pal is with the Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata 700108, India (e-mail: nrpal59@gmail.com).

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.2013.2255613

of a general type-2 fuzzy set, which reduces the computational overhead of a general type-2 fuzzy system significantly. For an IT2FS, the membership associated with an element is a subinterval of [0, 1]. In this paper, we use the interval type-2 fuzzy modeling to simplify the computational efforts to some extent. In [22]–[27], some interval type-2 FNNs are proposed for the designing of interval type-2 FLS. In [28]–[38], the authors have proposed automatic design of fuzzy rules, which are used in a variety of applications. A self-evolving interval type-2 fuzzy neural network (SEIT2FNN) is proposed in [30], which learns the structure and parameters in an online manner. The premise and consequent parameters in an SEIT2FNN are tuned by gradient descent and rule-ordered Kalman filter algorithm, respectively. The performances of SEIT2FNN are especially good for time-varying systems. Several interval type-2 FNNs [35]–[38], which use feedback/recurrent structure, are proposed for modeling of dynamic systems. In [35], a recurrent interval type-2 fuzzy neural network (RIT2FNN-A) that uses interval asymmetric type-2 fuzzy sets is proposed. This five-layer FNN uses a four-layer forward network and a feedback layer. In [36], the authors propose an internal/interconnection recurrent type-2 fuzzy neural network (IRT2FNN) structure that is suitable for dealing with time-varying systems. All free parameters of the IRT2FNN are updated via the gradient descent algorithm. Moreover, recurrent interval type-2 FNNs with local feedbacks are proposed in [37] and [38], where the recurrent property is achieved by locally feeding the firing strength of each rule back to itself. In [37], the consequent part of the recurrent self-evolving interval type-2 fuzzy neural network (RSEIT2FNN) is a linear function of current and past outputs and inputs. On the other hand, the consequent part in the RIFNN [38] is of Mamdani type, which is formulated as an interval-valued fuzzy set. In many papers, it has been seen that the TSK-type modeling can do an excellent job of modeling dynamic systems [9], [11], [30], [34], [36], [40].

This paper proposes a mutually recurrent interval type-2 neural fuzzy system (MRIT2NFS) for dynamic system identification. Our approach falls within the second category mentioned earlier, with an internal dynamics introduced in the consequent part. The MRIT2NFS has a self-evolving ability such that it can automatically evolve to acquire the required network structure as well as its parameters based on the training data. Therefore, to start the learning process, no preassigned network structure is necessary. In this proposed MRIT2NFS, we have three major contributions. First, we propose a novel recurrent NFS structure utilizing IT2FSs. Our network incorporates the advantage of local feedback and effective delivery of information through mutual feedbacks in the rule layer. In our network, the internal feedback and interaction loops in the antecedent part are formed by feeding the firing strength of each rule back to itself and to other rules. Second, we propose innovative learning algorithms for the structure and parameters of the system. Third, we also propose an interesting scheme to eliminate the less-useful recurrent weights. During the learning process, the MRIT2NFS may generate many recurrent weights when the rule base is bigger. As a result of elimination of the less-useful recurrent weights, our system achieves a significant reduction in both complexity and computational requirements. The consequent parameters in the MRIT2NFS are tuned by a rule-ordered Kalman filter algorithm. The antecedent parameters and all of the rule recurrent weights are learned by a gradient descent learning algorithm. To demonstrate the performance of MRIT2NFS, several simulations have been conducted. The performance of MRIT2NFS is also compared with that of recurrent type-1 FNNs, feedforward type-1 FNNs, and other type-2 FNNs.

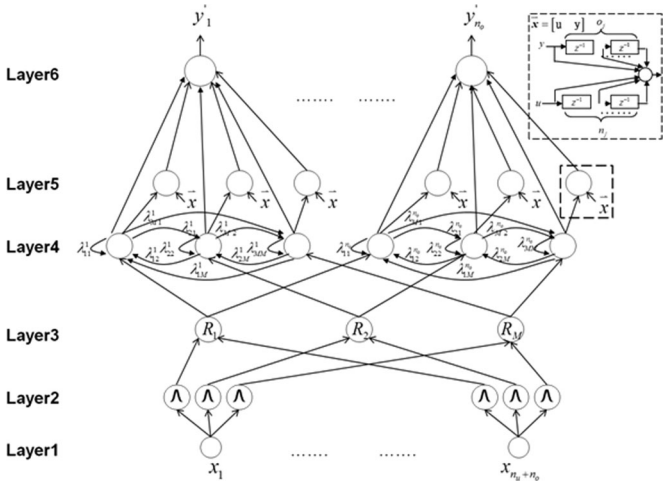

Fig. 1. Proposed six-layer MRIT2NFS structure, where each recurrent fuzzy rule in layer 4 forms an internal feedback and an interaction loop and each node in layer 5 performs a linear combination of current and lagged network inputs.

The rest of this paper is organized follows. Section II illustrates the MRIT2NFS structure. Section III discusses the structure and parameter learning methods for MRIT2NFS. Section IV presents results on five examples: one static system identification, three dynamic system identifications, and a chaotic time series prediction. Finally, Section V draws the conclusions.

II. Mutually Recurrent Interval Type-2 Neural Fuzzy System Structure

This section introduces the structure of an MRIT2NFS. This multi-input multi-output (MIMO) system consists of n_u inputs and n_o outputs. We represent the input and output of the dynamic system by $\boldsymbol{u}$ and $\mathbf{y}_p$, respectively, where $\boldsymbol{u} = (u_1, \ldots, u_{n_u})^T$ and $\boldsymbol{y}_p = (y_{p1}, \ldots, y_{pn_o})^T$. Fig. 1 shows the proposed six-layered MRIT2NFS structure. The detailed function of each layer is discussed next.

Layer 1 (Input layer): The inputs are crisp values. Only the current state $x(t) = (u(t), y_p(t))$ is fed as input to this layer. This is in contrast with the usual feedforward FNNs where both current and some past states are fed as inputs to input layer when such networks are used to model time-varying systems. To further clarify the inputs used in the system, we consider a system with one control input $u(t)$ and one system output $y(t)$. Then, at $t = 1$, $u(1)$ and $y(1)$ are used as inputs and the system output computed by the rules consequents is $y(2)$. Similarly, at $t = 2$, $u(2)$ and $y(2)$ are used as inputs and the computed system output is $y(3)$. Thus, current input and output both are used to define the rule antecedent. Note that this is a fan-out layer, and hence, there is no weight to be adjusted in this layer.

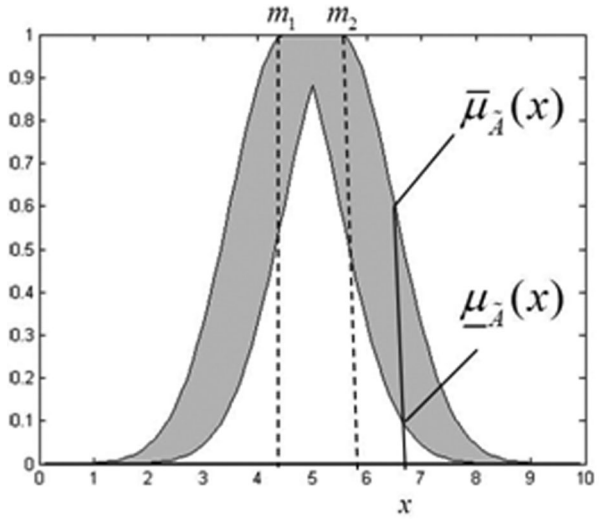

Fig. 2. IT2FS with Gaussian shape whose center (mean) is not known with certainty (the mean has an uncertainty). The mean can vary between m_1 and $m_2, m_1 < m_2$.

Layer 2 (MF layer): Each node in this layer performs fuzzification of one of the $(n_u + n_o)$ input variables using an interval type-2 MF, where n_u and n_o are the numbers of control inputs and system outputs. For the ith IT2FS $\tilde{A}_j^i$ on the input variable $x_j, j = 1, \ldots, (n_u + n_o)$, a Gaussian primary MF having a fixed standard deviation (STD) σ and an uncertain mean that takes on values in $[m_1, m_2]$ is used as shown in the following:

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

Fig. 2 depicts one such MF [20]. The uncertainty associated with the primary membership can be modeled in a more general manner. For example, we can consider uncertainty about the center of the MF as well as that about the spread of the MF. Here, following the protocol used in most type-2 neural fuzzy systems in the literature, we assumed that there is uncertainty only about the mean (the mean is not known with certainty) of the Gaussian MF. This choice yields an IT2FS, as depicted in Fig. 2.

The footprint of uncertainty [20] (the shaded region in Fig. 2) of this MF can be represented by the two bounding MFs: upper MF, $\bar{\mu}_j^i$ and lower MF, $\underline{\mu}_j^i$, where

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

and

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

Thus, the output of each node can be represented by an interval $[\underline{\mu}_j^i, \bar{\mu}_j^i]$ [20]. Another popular choice for type-2 MF is Trapezoidal MF. A general trapezoidal MF involves eight parameters to define it, while our Gaussian MF needs just three parameters and, thereby, drastically reduces the degrees of freedom (number of free variables) of the system. Moreover, the Gaussian function is differentiable, which helps us to use gradient-based tuning methods, and hence, we use it here.

Layer 3 (Spatial firing layer): Each node in this layer represents the antecedent part of a fuzzy rule, and it computes the spatial firing strength. We call it "spatial" as it does not depend on any temporal input, and also, we distinguish it from the "temporal" firing strength that is computed in the next layer. To compute the spatial firing strength F^i, each node performs a fuzzy meet operation on the inputs that it receives from layer 2 using an algebraic product operation. There are M nodes in this layer, where each node corresponds to one rule. Each node in this layer is connected to $n_u + n_o$ nodes of the previous layer. The structure learning process starts with no rule $(M = 0)$, and the first data point is used to generate the first rule making $M = 1$. Then, with new incoming data points, depending on how well a new data point matches with existing rules, new rules are generated. In Section III-A, we explain in detail how these M rules are generated. The spatial firing strength is an interval [13] and is computed as follows [27]:

$$F^i = [\underline{f}^i, \bar{f}^i], \quad i = 1, \ldots, M \tag{4}$$

$$\bar{f}^i = \prod_{j=1}^{n_u+n_o} \bar{\mu}_j^i, \ \underline{f}^i = \prod_{j=1}^{n_u+n_o} \underline{\mu}_j^i. \tag{5}$$

As explained earlier, in (4), M is the total number of rules. Use of (5) to compute the firing strength is probably the most common compared with the use of other T-norms [14], [21], [25], [26], [28]–[38]. Product is also used to compute rule firing strength even with type-1 Takagi–Sugeno-type systems [2], [4]–[12], [40]. A first look at (5) suggests that with a large number of antecedents, the firing strength will approach zero. However, this actually does not cause a problem for two reasons. The main reason is that the defuzzified output is computed as a convex combination of consequent values, where the weights of the convex combinations are the normalized temporal firing strength, which is computed using the firing strengths in (5). Moreover, for practical systems, the number of antecedent clauses involved in a rule is usually not very large. One can, of course, use minimum as the T-norm to compute the firing interval, but min is not differentiable (while product is), and hence, it is difficult to use the gradient-based tuning algorithm, which we use here. The problem associated with the use of minimum can be avoided by using a softer but differentiable version of minimum [43], [44], but this makes the learning rules quite complicated. Therefore, we restrict ourselves to product only.

Layer 4 (Temporal firing layer): There are $M \times n_o$ nodes in this layer. Each node in this layer is a recurrent rule node, which generates an internal feedback (self-loop) and external interconnection with mutual feedbacks. As a result, the recurrent weights λs are represented as self-loop and interconnection

weights. 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. In order to compute the temporal firing strength using (6), the nodes in this layer store the immediately past temporal firing strength (i.e., each node is equipped with some memory). Once the training starts, the initial value of the past temporal firing strength is set to zero.

The temporal firing strength $[\bar{\psi}_i^q(t), \underline{\psi}_i^q(t)], i = 1, \ldots, M$ and $q = 1, \ldots, n_o$, is computed combining the spatial firing strength $F^i(t)$ and previous temporal firing strength $\psi_i^q(t-1)$ using the following equation:

$$\psi_i^q(t) = \sum_{k=1} (\lambda_{ik}^q \cdot \psi_k^q(t-1)) + (1-\gamma_i^q) \cdot F^i(t) \quad (6)$$

where $\psi_i^q(t)$s lie in [0, 1], and $\lambda_{ik}^q = \frac{C_{ik}^q}{M}$ is the rule interaction weight between itself and other rules. Here, C_{ik}^q is the feedback related to node i of layer 4 from node k of layer 4 that is related to the qth output node. Thus, for local (internal) feedback $i = k$ and for i not equal to k, we get external feedback to the ith node of layer 4 from the kth rule (antecedents in layer 4) node. Assume that a new rule is generated, the initial value of C_{ik}^q in local feedback (i.e., for $i = k$), is set to 0.5, and for i not equal to k, C_{ik}^q is set to 0.2. The initial C_{ik}^q may be adjusted for different examples to yield better results. Thus, the total feedback to the ith rule node (layer 4) for the qth output node is $\gamma_i^q = \sum_{k=1}^M \lambda_{ik}^q$. The interval in (6) now may be written as

$$[\bar{\psi}_i^q(t), \underline{\psi}_i^q(t)] = \sum_{k=1}^{M} \lambda_{ik}^q \cdot [\bar{\psi}_k^q(t-1), \underline{\psi}_k^q(t-1)] + (1-\gamma_i^q) \cdot [\bar{f}^i(t), \underline{f}^i(t)] \quad (7)$$

where

$$\bar{\psi}_i^q(t) = \sum_{k=1} (\lambda_{ik}^q \cdot \bar{\psi}_k^q(t-1)) + (1-\gamma_i^q) \cdot \bar{f}^i(t) \quad (8)$$

and

$$\underline{\psi}_i^q(t) = \sum_{k=1} (\lambda_{ik}^q \cdot \underline{\psi}_k^q(t-1)) + (1-\gamma_i^q) \cdot \underline{f}^i(t). \quad (9)$$

Note that for the first epoch of the online learning, i.e., for $t = 1$, (6) will only use $(1-\gamma_i^q) \cdot F^i(t)$ because the initial past temporal firing strength is set to 0. For subsequent epochs, since the nodes in layer 4 store past firing strengths, (6) can be computed without any problem.

Layer 5 (Consequent layer): Each node in this layer is called a consequent node and functions as a linear model with exogenous inputs and lagged values. The output of a consequent node is a linear combination of current input states $x(t) = (u(t), y_p(t)) = (u_1(t), \ldots, u_{n_u}(t), y_{p1}(t), \ldots, y_{pn_o}(t))$ and their lagged values $(u_j(t-1), \ldots, u_j(t-N_j), y_{pk}(t-1), \ldots, y_{pk}(t-O_j))$. In Fig. 1, a consequent node is zoomed to elaborate its functioning. The output $\tilde{y}_q^i(t+1), i = 1, \ldots, M, q = 1, \ldots, n_o$, of the ith rule node connecting to the qth output variable is computed as follows:

$$\tilde{y}_q^i(t+1) = \sum_{j=0}^{n_u} \sum_{k=0}^{N_j} \tilde{a}_{jkq}^i u_j(t-k) + \sum_{j=1}^{n_o} \sum_{k=0}^{O_j} \tilde{a}_{(j+n_u)kq}^i y_{pj}(t-k) \quad (10)$$

where $u_0(t) = 1$ and $N_0 = 0$; N_j and O_j are the numbers of lagged control input $u_j(t)$ and system output $y_{pj}(t)$, respectively, and $\tilde{a}_{jkq}^i$s are interval valued coefficient denoted by

$$\tilde{a}_{jkq}^i = [c_{jkq}^i - s_{jkq}^i, c_{jkq}^i + s_{jkq}^i] \quad (11)$$

where c_{jkq}^i and s_{jkq}^i are the center and spread of an interval type-1 set, respectively. For an interval type-1 set, the membership value of every point over the interval $[c_{jkq}^i - s_{jkq}^i, c_{jkq}^i + s_{jkq}^i]$ is unity. Note that, in order to compute (10), some past values of u and y are needed, and these values are stored at appropriate nodes in this layer. In other words, nodes in this layer have some local memory. The inclusion of lagged values of $u(t)$ and $y_p(t)$ in the linear consequent part instead of the antecedent part simplifies the computation process of the network for modeling of dynamic systems, especially when IT2FSs 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 indices l and r denote left and right limits, respectively. According to (10) and (11), the node output is

$$\begin{aligned}\tilde{y}_q^i = [\tilde{y}_{lq}^i, \tilde{y}_{rq}^i] &= \sum_{j=0}^{n_u} \sum_{k=0}^{N_j} [c_{jkq}^i - s_{jkq}^i, c_{jkq}^i + s_{jkq}^i] \cdot u_j(t-k) \\ &+ \sum_{j=1}^{n_o} \sum_{k=0}^{O_j} [c_{(j+n_u)kq}^i - s_{(j+n_u)kq}^i, c_{(j+n_u)kq}^i + s_{(j+n_u)kq}^i] \\ &\cdot y_{pj}(t-k). \end{aligned} \quad (12)$$

That is

$$\begin{aligned}\tilde{y}_{lq}^i &= \sum_{j=0}^{n_u} \sum_{k=0}^{N_j} c_{jkq}^i u_j(t-k) + \sum_{j=1}^{n_o} \sum_{k=0}^{O_j} c_{(j+n_u)kq}^i y_{pj}(t-k) \\ &- \sum_{j=0}^{n_u} \sum_{k=0}^{N_j} s_{jkq}^i |u_j(t-k)| - \sum_{j=1}^{n_o} \sum_{k=0}^{O_j} s_{(j+n_u)kq}^i |y_{pj}(t-k)| \end{aligned} \quad (13)$$

and

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

Layer 6 (Output layer): Each node in this layer corresponds to one output variable. For defuzzification operation, the qth output layer node computes the network output variable y_q' using type-reduction. The type-reduced set is an interval set $[y_{lq}', y_{rq}']$. The outputs y_{lq}' and y_{rq}' can be computed using the Karnik–Mendel (KM) iterative procedure [13]. Using the KM procedure, we

shall rewrite the expressions for y'_{lq} and y'_{rq} of $[y'_{lq}, y'_{rq}]$ in suitable forms so that we can derive the learning rules easily. In the KM procedure, the consequent values are reordered in an ascending order. Let $\tilde{y}_{lq}$ and $\tilde{y}_{rq}$ be the original consequent values, and $\hat{y}_{lq}$ and $\hat{y}_{rq}$ be the corresponding rule ordered (in an ascending order) consequent values. Then, the relationship between $\tilde{y}_{lq}, \tilde{y}_{rq}, \hat{y}_{lq}$, and $\hat{y}_{rq}$ is

$$\hat{y}_{lq} = Q_l \tilde{y}_{lq} \text{ and } \hat{y}_{rq} = Q_r \tilde{y}_{rq}. \tag{15}$$

Here, Q_l and Q_r are $M \times M$ appropriate permutation matrices to reorder the values. Let $\bar{\psi}_q = (\bar{\psi}_q^1(t), \bar{\psi}_q^2(t) \ldots \bar{\psi}_q^M(t))^T$ and $\underline{\psi}_q = (\underline{\psi}_q^1(t), \underline{\psi}_q^2(t) \ldots \underline{\psi}_q^M(t))^T$. According to [27], the output y'_{lq} can be computed as follows:

$$\begin{aligned} y'_{lq} &= \frac{\sum_{i=1}^{L} (Q_l \bar{\psi}_q)_i \hat{y}^i_{lq} + \sum_{i=L+1}^{M} (Q_l \underline{\psi}_q)_i \hat{y}^i_{lq}}{\sum_{i=1}^{L} (Q_l \bar{\psi}_q)_i + \sum_{i=L+1}^{M} (Q_l \underline{\psi}_q)_i} \\ &= \frac{\bar{\psi}_q^T \mathbf{Q_l^T E_1^T E_1 Q_l \tilde{y}_{lq}} + \underline{\psi}_q^T \mathbf{Q_l^T E_2^T E_2 Q_l \tilde{y}_{lq}}}{\mathbf{p_l^T Q_l} \bar{\psi}_q + \mathbf{b_l^T Q_l} \underline{\psi}_q} \end{aligned} \tag{16}$$

where L and R denote the left and right crossover-over points, respectively. The end points L and R are defined in [13]. The other vectors and matrices involved in (16) are defined as

$$\begin{aligned} \mathbf{p_l} &= (\underbrace{1, 1, \ldots, 1}_{L}, 0, \ldots, 0)^T \in \Re^{M \times 1} \\ \mathbf{b_l} &= (0, \ldots, 0, \underbrace{1, \ldots, 1}_{M-L})^T \in \Re^{M \times 1} \end{aligned} \tag{17}$$

$$\begin{aligned} \mathbf{E_1} &= (e_1, e_2, \ldots, e_L, 0, \ldots, 0) \in \Re^{L \times M}, \text{ and} \\ \mathbf{E_2} &= (0, \ldots, 0, \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{M-L}) \in \Re^{(M-L) \times M} \end{aligned} \tag{18}$$

where $\boldsymbol{e}_i \in \Re^{L \times 1}$ and $\boldsymbol{\varepsilon}_i \in \Re^{M-L}$ are unit vectors, whose all but ith element is zero and the ith element is one. In (16), $(Q_l \bar{\psi}_q)$ produces a vector in M dimension where the components of $(Q_l \bar{\psi}_q)$ are permuted version of components in $\bar{\psi}_q$. In addition, $(Q_l \bar{\psi}_q)_i$ represents the ith component of $(Q_l \bar{\psi}_q)$. Thus, (19) computes a convex combination of $\bar{y}^i_{rq}, i = 1, \ldots, M$, values. The weights of the convex combination are computed from the components of $(Q_l \bar{\psi}_q)$, which are nothing but the temporal firing strength in (8).

Similarly, the output y'_{rq} can be computed as follows:

$$\begin{aligned} y'_{rq} &= \frac{\sum_{i=1}^{R} (Q_r \underline{\psi}_q)_i \hat{y}^i_{rq} + \sum_{i=R+1}^{M} (Q_r \bar{\psi}_q)_i \hat{y}^i_{rq}}{\sum_{i=1}^{R} (Q_r \underline{\psi}_q)_i + \sum_{i=R+1}^{M} (Q_r \bar{\psi}_q)_i} \\ &= \frac{\underline{\psi}_q^T \mathbf{Q_r^T E_3^T E_3 Q_r \tilde{y}_{rq}} + \bar{\psi}_q^T \mathbf{Q_r^T E_4^T E_4 Q_r \tilde{y}_{rq}}}{\mathbf{p_r^T Q_r} \underline{\psi}_q + \mathbf{b_r^T Q_r} \bar{\psi}_q} \end{aligned} \tag{19}$$

where

$$\begin{aligned} \boldsymbol{p_r} &= (\underbrace{1, 1, \ldots, 1}_{R}, 0, \ldots 0)^T \in \Re^{M \times 1} \\ \boldsymbol{b_r} &= (0, \ldots, 0, \underbrace{1, \ldots, 1}_{M-R})^T \in \Re^{M \times 1} \end{aligned} \tag{20}$$

$$\begin{aligned} \mathbf{E_3} &= (\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_R, \mathbf{0}, \ldots, \mathbf{0}) \in \Re^{R \times M} \\ \mathbf{E_4} &= (\mathbf{0}, \ldots, \mathbf{0}, \boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \ldots, \boldsymbol{\varepsilon}_{M-R}) \in \Re^{(M-R) \times M} \end{aligned} \tag{21}$$

and where $\boldsymbol{e}_i \in \Re^{R \times 1}$ and $\boldsymbol{\varepsilon}_i \in \Re^{M-R}$ are unit vectors (all but ith element are zero and the ith element is 1). All of unit vectors are defined in [27]. Such an 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

$$y'_q = \frac{y'_{lq} + y'_{rq}}{2}. \tag{22}$$

III. Mutually Recurrent Interval Type-2 Neural Fuzzy System Learning

Initially, there is no rule in an MRIT2NFS. All of the recurrent type-2 fuzzy rules evolve from simultaneous structure and parameter learning. The following sections introduce the structure and parameter learning algorithm explicitly.

A. Structure Learning

The online rule is generated according to the structure learning algorithm. A previous study [30] utilized the rule firing strength as a criterion for type-1 fuzzy rule generation. This idea is extended to type-2 fuzzy rule generation using an MRIT2NFS. The spatial firing strength F^i in (4) is used to decide whether a new rule should be generated. The type-2 rule firing strength is an interval. The center of the spatial firing interval, i.e., $f_c = \frac{1}{2}(\bar{f}^i + \underline{f}^i)$, is used as a criterion for rule generation. The first incoming data point $\mathbf{x}$ is used to generate the first fuzzy rule, and the uncertain mean and width of the type-2 fuzzy MFs associated with this rule are set as

$$\begin{aligned} \left[m^1_{j1}, m^1_{j2}\right] &= [x_j - 0.1, x_j + 0.1] \text{ and } \sigma = \sigma_{\text{fixed}} \\ & j = 1, \ldots, n_u + n_o \end{aligned} \tag{23}$$

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. Subsequently, for each of new incoming data $\mathbf{x}(t)$, we find

$$I = \arg \max_{1 \le i \le M(t)} f^i_c(t) \tag{24}$$

where $M(t)$ is the number of existing rules at time t. If $f^I_c(t) \le f_{\text{th}} (0 < f_{\text{th}} \le 1$ is a prespecified threshold), then a new fuzzy rule is generated. *A smaller value f_{th} generates a smaller number of rules. Conversely, a large number of rules may be generated when f_{th} is set to be a high value.* The idea is that if the present data point does not match well with any of the existing rules, then a new rule is generated. Here also, we use the same procedure to assign the uncertain mean as done for the very first rule, i.e., for the new rule, the means of the corresponding type-2 fuzzy sets are defined as

$$\begin{aligned} \left[m^{M(t)+1}_{j1}, m^{M(t)+1}_{j2}\right] &= [x_j(t) - 0.1, x_j(t) + 0.1] \\ & j = 1, \ldots, n_u + n_o. \end{aligned} \tag{25}$$

The width of the each fuzzy set associated with a new rule is defined as follows:

$$\sigma^{M(t)+1} = \beta \cdot \left(\sum_{j=1}^{n_u+n_o} \left(x_j - \frac{m_{j1}^I + m_{j2}^I}{2} \right)^2 \right)^{\frac{1}{2}}. \quad (26)$$

In (26), I is the index of the best matching rule, and m_{j1}^I and m_{j2}^I are the means of the MF of the jth antecedent clause of the Ith rule. Equations (23) and (25) indicate that for the mean, the width of the uncertain region is 0.2. If the uncertainty associated with the mean is made too small, then the type-2 fuzzy sets become similar to type-1 fuzzy sets. On the other hand, if the width of the uncertain region is too large, then the uncertain mean covers most of input domain. Equation (26) indicates that the initial width is equal to the Euclidean distance between current input data $\mathbf{x}$ and the center of best matching rule for this data point times an overlap degree $\beta(\beta > 0)$. *If β is set to a very large value, then it may result in a high overlap between a fuzzy set associated with the new and fuzzy sets associated with the existing rules, while with a very small value of β, practically, there will be no overlap between fuzzy sets.* In this study, β is set to 0.5 so that the width of new type-2 fuzzy set is half of the Euclidean distance from the center of the best matching rule, and an adequate overlap between adjacent rules is realized.

B. Parameter Learning

Along with the learning of the structure, the parameters are also learned. All free parameters of the MRIT2NFS are adjusted with each incoming training data regardless of whether a rule is generated. For clarity, let us just consider the qth output of the network. The parameter learning process updates the network parameters minimizing the error function

$$E = \frac{1}{2}[y_q'(t+1) - y_d(t+1)]^2 \quad (27)$$

where $y_q'(t+1)$ and $y_d(t+1)$ represent the MRIT2NFS output and the desired output, respectively. The parameters in the consequent part are learned based on the rule-ordered Kalman filter algorithm [30], as described next. To compute y_{lq}' and y_{rq}' in the KM iterative procedure, the required precondition is that $\tilde{y}_{lq}$ and $\tilde{y}_{rq}$ are rearranged in an ascending order. As the consequent values $\tilde{y}_{lq}$ and $\tilde{y}_{rq}$ change, their rule-ordering may also change. The (15) indicated the arranged consequent values with respect to the original rule order. According to the mapping, (16) and (19) are expressed by $\tilde{y}_{lq}$ and $\tilde{y}_{rq}$ as follows [37]:

$$y_{lq}' = \phi_{lq}^T \tilde{\mathbf{y}}_{\mathbf{lq}}$$
$$\phi_{lq} = \frac{\bar{\psi}_q^T \mathbf{Q}_\mathbf{l}^\mathbf{T} \mathbf{E}_\mathbf{1}^\mathbf{T} \mathbf{E}_\mathbf{1} \mathbf{Q}_\mathbf{l} + \underline{\psi}_q^T \mathbf{Q}_\mathbf{l}^\mathbf{T} \mathbf{E}_\mathbf{2}^\mathbf{T} \mathbf{E}_\mathbf{2} \mathbf{Q}_\mathbf{l}}{\mathbf{p}_\mathbf{l}^\mathbf{T} \mathbf{Q}_\mathbf{l} \bar{\psi}_q + \mathbf{b}_\mathbf{l}^\mathbf{T} \mathbf{Q}_\mathbf{l} \underline{\psi}_q} \in \Re^{M \times 1} \quad (28)$$

and

$$y_{rq}' = \phi_{rq}^T \tilde{\mathbf{y}}_{\mathbf{rq}}$$
$$\phi_{rq} = \frac{{}^T_q \mathbf{Q}_\mathbf{r}^\mathbf{T} \mathbf{E}_\mathbf{3}^\mathbf{T} \mathbf{E}_\mathbf{3} \mathbf{Q}_\mathbf{r} + \bar{\psi}_q^T \mathbf{Q}_\mathbf{r}^\mathbf{T} \mathbf{E}_\mathbf{4}^\mathbf{T} \mathbf{E}_\mathbf{4} Q_r}{\mathbf{p}_\mathbf{r}^\mathbf{T} \mathbf{Q}_\mathbf{r} \underline{\psi}_q + \mathbf{b}_\mathbf{r}^\mathbf{T} \mathbf{Q}_\mathbf{r} \bar{\psi}_q} \in \Re^{M \times 1}. \quad (29)$$

Thus, the output y_q' in (22) can be re-expressed as

$$y_q' = \frac{1}{2}(y_{lq}' + y_{rq}') = \frac{1}{2}(\phi_{lq}^T \tilde{\boldsymbol{y}}_{lq} + \phi_{rq}^T \tilde{\boldsymbol{y}}_{rq})$$
$$= [\bar{\phi}_{lq}^T \ \bar{\phi}_{rq}^T] \begin{bmatrix} \tilde{\boldsymbol{y}}_{lq} \\ \tilde{\boldsymbol{y}}_{rq} \end{bmatrix} = [\bar{\phi}_{lq}^1 \cdots \bar{\phi}_{lq}^M \ \bar{\phi}_{rq}^1 \cdots \bar{\phi}_{rq}^M] \begin{bmatrix} \tilde{\boldsymbol{y}}_{lq}^1 \\ \vdots \\ \tilde{\boldsymbol{y}}_{lq}^M \\ \tilde{\boldsymbol{y}}_{rq}^1 \\ \vdots \\ \tilde{\boldsymbol{y}}_{rq}^M \end{bmatrix} \quad (30)$$

where $\bar{\phi}_{lq}^T = 0.5\phi_{lq}^T$ and $\bar{\phi}_r \underline{\psi}_q^T = 0.5\phi_{rq}^T$. According to (13) and (14), (30) can be further expressed as (31), shown at the bottom of the page [37].

In (31), c_{jkq}^l and s_{jkq}^l are the coefficients of the linear equations involved in the consequents. During the online structure learning, the dimension of $\tilde{\boldsymbol{y}}_{lq}$ and $\tilde{\boldsymbol{y}}_{rq}$ increases with time, and the positions of c_{jkq} and s_{jkq} change accordingly within the same vector. To keep the position of c_{jkq} and s_{jkq} unaltered in the vector, the rule-ordered Kalman filtering algorithm rearranges elements in rule order in (31). Let $\tilde{\boldsymbol{v}}_{\mathrm{TSK}}$ denote the

$$y_q' = [\bar{\phi}_{lq}^T \bar{\phi}_{rq}^T] \begin{bmatrix} \tilde{\boldsymbol{y}}_{lq} \\ \tilde{\boldsymbol{y}}_{rq} \end{bmatrix} = [\bar{\phi}_{lq}^1 \cdots \bar{\phi}_{lq}^M \ \bar{\phi}_{rq}^1 \cdots \bar{\phi}_{rq}^M]$$
$$\begin{bmatrix} \sum_{j=0}^{n_u} \sum_{k=0}^{N_j} c_{jkq}^1 u_j(t-k) + \sum_{j=1}^{n_o} \sum_{k=0}^{O_j} c_{(j+n_u)kq}^1 y_{pj}(t-k) - \sum_{j=0}^{n_u} \sum_{k=0}^{N_j} s_{jkq}^1 |u_j(t-k)| - \sum_{j=1}^{n_o} \sum_{k=0}^{O_j} s_{(j+n_u)kq}^1 |y_{pj}(t-k)| \\ \vdots \\ \sum_{j=0}^{n_u} \sum_{k=0}^{N_j} c_{jkq}^M u_j(t-k) + \sum_{j=1}^{n_o} \sum_{k=0}^{O_j} c_{(j+n_u)kq}^M y_{pj}(t-k) - \sum_{j=0}^{n_u} \sum_{k=0}^{N_j} s_{jkq}^M |u_j(t-k)| - \sum_{j=1}^{n_o} \sum_{k=0}^{O_j} s_{(j+n_u)kq}^M |y_{pj}(t-k)| \\ \sum_{j=0}^{n_u} \sum_{k=0}^{N_j} c_{jkq}^1 u_j(t-k) + \sum_{j=1}^{n_o} \sum_{k=0}^{O_j} c_{(j+n_u)kq}^1 y_{pj}(t-k) + \sum_{j=0}^{n_u} \sum_{k=0}^{N_j} s_{jkq}^1 |u_j(t-k)| + \sum_{j=1}^{n_o} \sum_{k=0}^{O_j} s_{(j+n_u)kq}^1 |y_{pj}(t-k)| \\ \vdots \\ \sum_{j=0}^{n_u} \sum_{k=0}^{N_j} c_{jkq}^M u_j(t-k) + \sum_{j=1}^{n_o} \sum_{k=0}^{O_j} c_{(j+n_u)kq}^M y_{pj}(t-k) + \sum_{j=0}^{n_u} \sum_{k=0}^{N_j} s_{jkq}^M |u_j(t-k)| + \sum_{j=1}^{n_o} \sum_{k=0}^{O_j} s_{(j+n_u)kq}^M |y_{pj}(t-k)| \end{bmatrix}. \quad (31)$$

vector all consequent parameters, i.e.,

$$\tilde{\mathbf{v}}_{TSK} = [c^1_{00q} \dots c^1_{(n_o+n_u)N_{n_o}q} s^1_{00q} \dots s^1_{(n_o+n_u)N_{n_o}q} \dots \times c^M_{00q} \dots c^M_{(n_o+n_u)N_{n_o}q} s^M_{00q} \dots s^M_{(n_o+n_u)N_{n_o}q}]^T \quad (32)$$

where the consequent parameters are placed according to the rule order. Equation (31) can now be re-expressed as

$$\begin{aligned} y'_q &= [\bar{\phi}_{c1} u_0 \dots \bar{\phi}_{c1} y_{pn_o}(t-O_{n_o}) - \bar{\phi}_{s1}|u_0| \dots - \bar{\phi}_{s1}|y_{pn_o} \times (t-O_{n_o})| \dots \dots \bar{\phi}_{cM} u_0 \dots \bar{\phi}_{cM} y_{pn_o}(t-O_{n_o}) - \bar{\phi}_{sM}|u_0| \dots - \bar{\phi}_{sM}|y_{pn_o}(t-O_{n_o})|]\tilde{\mathbf{v}}_{\mathrm{TSK}} \\ &= \bar{\boldsymbol{\phi}}'^{T}_{\mathrm{TSK}} \tilde{\mathbf{v}}_{\mathrm{TSK}} \end{aligned} \quad (33)$$

where $\bar{\phi}^j_{cq} = \bar{\phi}^j_{lq} + \bar{\phi}^j_{rq}$ and $\bar{\phi}^j_{sq} = \bar{\phi}^j_{rq} - \bar{\phi}^j_{lq}, j = 1, \dots, M$. The consequent parameter vector $\tilde{\mathbf{v}}_{\mathrm{TSK}}$ is updated by executing the following rule-ordered Kalman filtering algorithm [37]:

$$\begin{aligned} \tilde{\mathbf{v}}_{\mathrm{TSK}}(\mathbf{t+1}) &= \tilde{\mathbf{v}}_{\mathrm{TSK}}(\mathbf{t}) + \mathbf{S}(\mathbf{t+1})\bar{\boldsymbol{\phi}}'_{\mathrm{TSK}}(\mathbf{t+1})(y^d(t+1) - \bar{\boldsymbol{\phi}}'^{\mathrm{T}}_{\mathrm{TSK}}(\mathbf{t+1})\tilde{\mathbf{v}}_{\mathrm{TSK}}(\mathbf{t})) \\ \mathbf{S}(\mathbf{t+1}) &= \frac{1}{\kappa}\left[\mathbf{S}(\mathbf{t}) - \frac{\mathbf{S}(\mathbf{t})\bar{\boldsymbol{\phi}}'_{\mathrm{TSK}}(\mathbf{t+1})\bar{\boldsymbol{\phi}}'^{\mathrm{T}}_{\mathrm{TSK}}(\mathbf{t+1})\mathbf{S}(\mathbf{t})}{\kappa + \bar{\boldsymbol{\phi}}'^{\mathrm{T}}_{\mathrm{TSK}}(\mathbf{t+1})\mathbf{S}(\mathbf{t})\bar{\boldsymbol{\phi}}'_{\mathrm{TSK}}}\right] \end{aligned} \quad (34)$$

where $0 < \kappa \leq 1$ is a forgetting factor. (Just to keep parity with a previous study [37], we have used $\kappa = 0.99995$; however, our experience suggests that one can use $\kappa = 1.0$ without much effect, as expected, on the overall performance.) The dimension of $\tilde{\mathbf{v}}_{\mathrm{TSK}}$ and $\bar{\boldsymbol{\phi}}'_{\mathrm{TSK}}$ and the matrix S increases when a new rule is generated. When a new rule evolves, MRIT2NFS augments $\mathbf{S}(\mathbf{t})$ as follows, (35), shown at the bottom of the page, where C is a large positive constant (we use $C = 10$), and the size of the identity matrix $\mathbf{I}$ is

$$2\left(\sum_{j=0}^{n_u}(N_j+1) + \sum_{j=1}^{n_o}(O_j+1)\right) \times 2\left(\sum_{j=0}^{n_u}(N_j+1) + \sum_{j=1}^{n_o}(O_j+1)\right). \quad (36)$$

Note that $S(0)$ is 1×1 matrix. We used $S(0) = [10]$. The antecedent parameters of MRIT2NFS are tuned by the gradient descent algorithm. For convenience of notations for the gradient descent learning rules, according to [27], (16) can be rewritten as

$$y'_{lq} = \frac{\bar{\boldsymbol{\psi}}^T_q \boldsymbol{a}_{lq} + \underline{\boldsymbol{\psi}}^T_q \boldsymbol{b}_{lq}}{\bar{\boldsymbol{\psi}}^T_q \boldsymbol{c}_{lq} + \underline{\boldsymbol{\psi}}^T_q \boldsymbol{d}_{lq}} \quad (37)$$

where

$$\begin{aligned} \mathbf{a_{lq}} &= \mathbf{Q_l^T E_1^T E_1 Q_l \tilde{y}_{lq}} \in \Re^{M\times 1} \\ \mathbf{b_{lq}} &= \mathbf{Q_l^T E_2^T E_2 Q_l \tilde{y}_{lq}} \in \Re^{M\times 1} \end{aligned} \quad (38)$$

$$\mathbf{c_{lq}} = \mathbf{Q_l^T p_l} \in \Re^{M\times 1},\ \mathbf{d_{lq}} = \mathbf{Q_l^T b_l} \in \Re^{M\times 1}. \quad (39)$$

Similarly, (19) can be rewritten as

$$y'_{rq} = \frac{\underline{\boldsymbol{\psi}}^T_q \boldsymbol{a}_{rq} + \bar{\boldsymbol{\psi}}^T_q \boldsymbol{b}_{rq}}{\underline{\boldsymbol{\psi}}^T_q \boldsymbol{c}_{rq} + \bar{\boldsymbol{\psi}}^T_q \boldsymbol{d}_{rq}} \quad (40)$$

where

$$\begin{aligned} \mathbf{a_{rq}} &= \mathbf{Q_r^T E_3^T E_3 Q_r \tilde{y}_{rq}} \in \Re^{M\times 1} \\ \mathbf{b_{rq}} &= \mathbf{Q_r^T E_4^T E_4 Q_r \tilde{y}_{rq}} \in \Re^{M\times 1} \end{aligned} \quad (41)$$

$$\mathbf{c_{rq}} = \mathbf{Q_r^T p_r} \in \Re^{M\times 1},\ \mathbf{d_{rq}} = \mathbf{Q_r^T b_r} \in \Re^{M\times 1}. \quad (42)$$

Using the gradient descent algorithm, we have

$$\lambda^q_{ik}(t+1) = \lambda^q_{ik}(t) - \eta \frac{\partial E}{\partial \lambda^q_{ik}(t)} \quad (43)$$

where η is a learning constant ($\eta = 0.075$ in this paper) and

$$\frac{\partial E}{\partial \lambda^q_{ik}} = \frac{\partial E}{\partial y'_q}\left(\frac{\partial y'_q}{\partial y'_{lq}}\frac{\partial y'_{lq}}{\partial \lambda^q_{ik}} + \frac{\partial y'_q}{\partial y'_{rq}}\frac{\partial y'_{rq}}{\partial \lambda^q_{ik}}\right) = \frac{1}{2}(y'_q - y_d) \times \left[\left(\frac{\partial y'_{lq}}{\partial \bar{\psi}^q_i} + \frac{\partial y'_{rq}}{\partial \bar{\psi}^q_i}\right)\frac{\partial \bar{\psi}^q_i}{\partial \lambda^q_{ik}} + \left(\frac{\partial y'_{lq}}{\partial \underline{\psi}^q_i} + \frac{\partial y'_{rq}}{\partial \underline{\psi}^q_i}\right)\frac{\partial \underline{\psi}^q_i}{\partial \lambda^q_{ik}}\right] \quad (44)$$

where

$$\frac{\partial y'_{lq}}{\partial \bar{\psi}^q_i} = \frac{a_{lqi} - y'_{lq} c_{lqi}}{\bar{\psi}^T_q \boldsymbol{c}_{lq} + \underline{\psi}^T_q \boldsymbol{d}_{lq}},\ \frac{\partial y'_{rq}}{\partial \bar{\psi}^q_i} = \frac{b_{rqi} - y'_{rq} d_{rqi}}{\underline{\psi}^T_q \boldsymbol{c}_{rq} + \bar{\psi}^T_q \boldsymbol{d}_{rq}} \quad (45)$$

$$\frac{\partial y'_{lq}}{\partial \underline{\psi}^q_i} = \frac{b_{lqi} - y'_{lq} d_{lqi}}{\bar{\psi}^T_q \boldsymbol{c}_{lq} + \underline{\psi}^T_q \boldsymbol{d}_{lq}},\ \frac{\partial y'_{rq}}{\partial \underline{\psi}^q_i} = \frac{a_{rqi} - y'_{rq} c_{rqi}}{\underline{\psi}^T_q \boldsymbol{c}_{rq} + \bar{\psi}^T_q \boldsymbol{d}_{rq}} \quad (46)$$

$$\frac{\partial \bar{\psi}^q_i}{\partial \lambda^q_{ik}} = \bar{\psi}^q_k(t-1) - \bar{f}^i(t),\ \frac{\partial \underline{\psi}^q_i}{\partial \lambda^q_{ik}} = \underline{\psi}^q_k(t-1) - f^i(t). \quad (47)$$

Details of the learning equations for parameters, including m_{j1}, m_{j2}, and σ, of the antecedents can be found in [27].

Pruning of less-important rules: Our system involves many recurrent weights. Depending on the nature of the underlying system that we are trying to model, all of these recurrent feedbacks may not be important. In fact, if the magnitude of a recurrent weight is very low, then that weight will not have much effect on the system output, and hence, such weights/feedbacks can be dropped. This is what we do. If the absolute value of a recurrent weight is less than a predefined threshold ε, we delete that connection ε lies in [0,1]. Once we delete some recurrent weights (feedback connections), we must adapt the system in its new environment, and hence, we retrain the network a few epochs (here, we use only five epochs). The MRIT2NFS that uses such elimination of recurrent weights is called MRIT2NFS-ε.

$$\mathbf{S}(\mathbf{t}) = \text{block diag}[\mathbf{S}(\mathbf{t})C \cdot I] \in \Re^{2(M+1)(\sum_{j=0}^{n_u}(N_j+1)+\sum_{j=1}^{n_o}(O_j+1))\times 2(M+1)(\sum_{j=0}^{n_u}(N_j+1)+\sum_{j=1}^{n_o}(O_j+1))} \quad (35)$$

TABLE I
LIST OF PARAMETERS USED IN MRIT2NFS

Symbol	Description	Constraint on its values
f_{th}	Structure threshold of rule granularity	Any value in [0, 1]
β	Controls the degree of overlap between type-2 fuzzy sets for a new rule with those of the existing rules in rule generation	Any value in [0, 1]
ε	Threshold for elimination of Recurrent weights	Any value in [0, 1]
C_{ik}^{q}	Used to define the rule interaction weight	Any value in [0, 1]
N_1	Maximum delays of the control input	Depends on the example
O_1	Maximum delays of the system output	Depends on the example

IV. SIMULATION

Our MRIT2NFS uses a set of parameters. For convenience, we have summarized these parameters in Table I. Next, we describe application of MRIT2NFS on five problems. These examples include identification of two single-input single-output (SISO) dynamic systems (see Examples 1 and 2), one MIMO dynamic system (see Example 3), prediction of chaotic time series (see Example 4), and identification of nonlinear system plant (see Example 5). For all but Example 5, we normalize the datasets in $[-1, 1]$. Example 5 is not a dynamical system, and it is comparatively easy to learn. Therefore, we did not normalize the data, but we have used the same membership definition as in (23). We shall see later that even in this case, the performance of the system is very satisfactory indicating the robustness of our system. The performance of MRIT2NFS is compared with that of recurrent and feedback type-1 and type-2 FNNs.

A. Example 1 (Dynamic System Identification)

This example uses MRIT2NFS to identify an SISO linear time-varying system, which was introduced in [9]. The dynamic system with lagged inputs is guided by the following difference equation:

$$y_p(t+1) = f(y_p(t), y_p(t-1), y_p(t-2), u(t), u(t-1)) \tag{48}$$

where

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) x_4}{1 + x_2^2 + x_3^2}. \tag{49}$$

The system has a single input (i.e., $n_u = 1$) and a single output (i.e., $n_o = 1$). The current variables $u(t)$ and $y_p(t)$ are fed as inputs to the MRIT2NFS input layer. The current output of the plant depends on two previous outputs and one previous input. Therefore, the consequent part parameters of MRIT2NFS are set as $N_1 = 2$ and $O_1 = 1$. The training procedure minimizes the square error between the output of the system $y_p(t+1)$ and the target output $y_d(t+1)$. To train the MRIT2NFS, we follow the same computational protocols as in [9], i.e., we use only ten epochs and there are 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 defined by $1.05\sin(\pi t/45)$. This type of training is analogous to an online training process, where the total number of online training time steps is 9000. The structure learning threshold f_{th} influences the number of fuzzy rules to be generated. After training, two recurrent fuzzy rules are generated when f_{th} is set to 0.02. Table II shows the root-mean-squared error (RMSE) on the training data. To validate the identified system, as adopted in [9], we use the following input:

$$u(t) = \begin{cases} \sin\left(\dfrac{\pi t}{25}\right), & t < 250 \\ 1.0, & 250 \le t < 500 \\ -1.0, & 500 \le t < 750 \\ 0.3\sin\left(\dfrac{\pi t}{25}\right) + 0.1\sin\left(\dfrac{\pi t}{32}\right) & \\ \quad +0.6\sin\left(\dfrac{\pi t}{10}\right), & 750 \le t < 1000. \end{cases} \tag{50}$$

Fig. 3 compares the actual output with the output produced by MRIT2NFS for the test input generated using (50), while Fig. 4 shows the test error between the desired output and actual output produced by MRIT2NFS. Figs. 3 and 4 together reveal a very good match suggesting that MRIT2NFS architecture along with our system identification scheme does a very good job of identifying the dynamical system with feedback.

Table II compares the performance of MRIT2NFS with that of seven different approaches including TSK-type feedforward type-1 and type-2 FNNs, a recurrent NN, and type-1 recurrent FNNs. The comparison is done in terms of number of rules, number of free parameters, training RMSE, and test RMSE. As in MRIT2NFS, all these networks use the same information including number of input variables, training data, test data, and training epochs. In order to make a fair comparison, the total number of parameters of the feedforward type-1 FNN is kept similar to that of feedforward interval type-2 FNN. The number of parameters in an interval type-2 FNN is larger than that in a feedforward type-1 FNN because of extra free parameters in type-2 fuzzy sets and rule consequent part. Consequently, the number of rules used in a feedforward type-1 FNN is larger than that in a feedforward interval type-2 FNN, as shown in Table I. The compared feedforward type-2 FNN is an interval type-2 FNN with uncertain means, where all network parameters are learned by the gradient descent algorithm. Our result reveals the advantage of using recurrent structure in the MRIT2NFS, which achieves smaller test RMSE than that by the feedforward type-2 FNN. All of the compared recurrent type-2 FNNs use the same fuzzy sets in the antecedent part, i.e., use IT2FSs with uncertain means. The performance of MRIT2NFS is also compared with other interval type-2 FNNs with recurrent structure, including a recurrent self-evolving interval type-2 fuzzy neural network with uncertain means (RSEIT2FNN-UM) [37] and a recurrent interval-valued fuzzy neural network (RIFNN) [38]. In these cases also, MRIT2NFS yields better test accuracies. However, RIFNN and RSEIT2FNN-UM use a marginally lower number of free parameters. Our results demonstrate that MRIT2NFS can effectively capture information about the system using mutual feedbacks and outperforms the RSEIT2FNN, which only uses local feedbacks. The recurrent type-1 FNNs considered include the wavelet-based RFNN (WRFNN) [6], TSK-type

TABLE II
PERFORMANCE OF MRIT2NFS AND OTHER RECURRENT MODELS FOR SISO PLANT IDENTIFICATION IN EXAMPLE 1

Models	Feedforward Type-1 FNN	WRFNN [6]	TRFN-S [9]	RSEFNN [11]	Feedforward Type-2 FNN	RSEIT2FNN -UM[37]	RIFNN [38]	MRIT2NFS
Number of Rules	7	5	5	4	4	2	4	2
Number of Parameters	42	55	60	32	48	38	36	40
Training RMSE	0.0178	0.064	0.021	0.02	0.032	0.0048	0.023	**0.0008**
Test RMSE	0.0528	0.098	0.041	0.0397	0.052	0.049	0.0465	**0.0078**

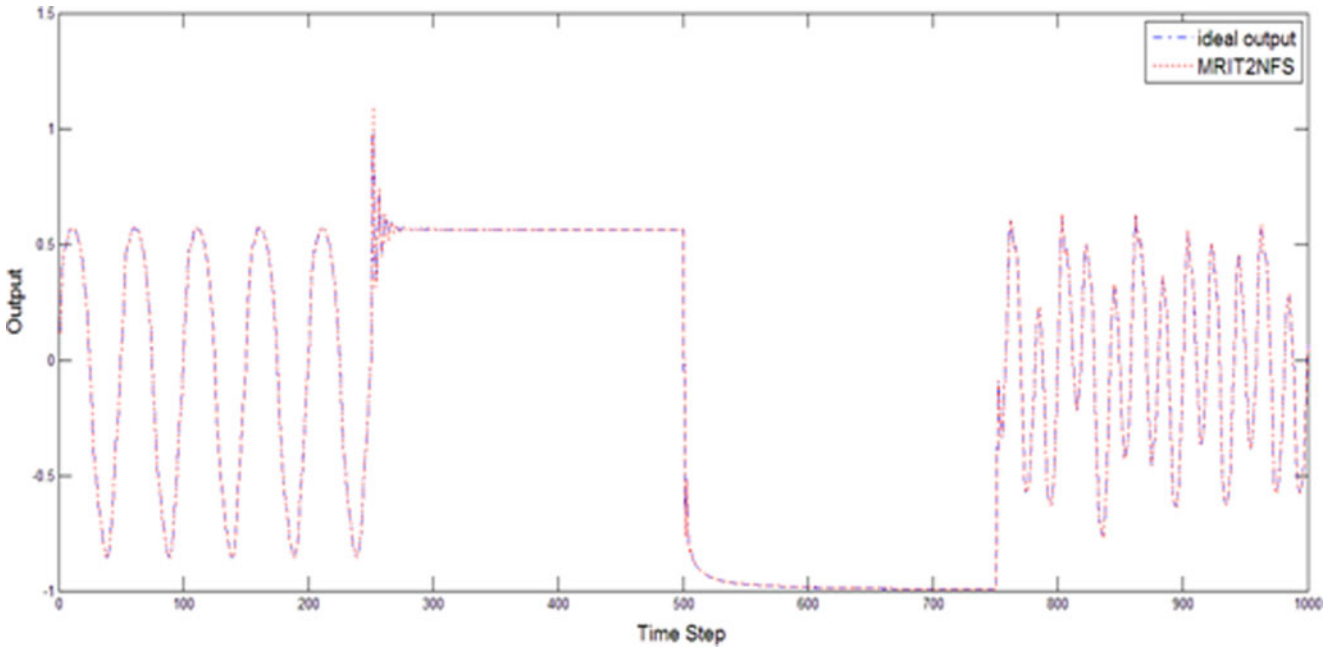

Fig. 3. Outputs of the dynamic plant (dashed-dotted line) and MRIT2NFS (dotted line) in Example 1.

recurrent fuzzy network with supervised learning (TRFN-S) [9], and RSEFNN-LF [11]. Table I indicates that the test error of the TRFN and the RIFNN are very close for the noise-free environment, but for noisy environment, the TRFN-S is found to perform better. We have also compared the performance between TRFN and RSEIT2FNN. The recurrent structure with only local feedbacks in RSEIT2FNN may not be adequate for this example because the rules lack the information from other rules. For this reason, the performance of the TRFN is found to be better than that of the RSEIT2FNN (see Table II).

Like any other type-2 method, the proposed type-2 methods demand more computation, but it can yield more quality outputs. Moreover, although each learning step is computationally more expensive compared with that of its type-1 counterpart, our network, using the same number of iterations, yields a better solution (faster convergence) than the type-1 systems.

There are a few parameters, i.e., $f_{\rm th}, \beta$, and ε, that are involved in the learning of MRIT2NFS. We now investigate the influence of these parameters on the performance of MRIT2NFS. In addition, we shall also consider the robustness of the system against noise in the inputs. The threshold parameter $f_{\rm th}$ decides the number of rules in the MRIT2NFS, while the parameter ε in MRIT2NFS-ε is used to decide the feedback connections in layer 4 that could be removed. Table III shows the MRIT2NFS performance for different values of $f_{\rm th}$ and ε when $\beta = 0.5$.

As expected, larger values of $f_{\rm th}$ result in larger numbers of rules, and larger ε reduces the number of tunable parameters in the system. Table II suggests that although different choices of $f_{\rm th}$ and ε change the number of rules marginally, the training and test errors practically do not change. Thus, at least for this dataset, our system is quite robust with respect to the choice of these two parameters. From a user point of view, the network with the smallest number of free parameters that can provide the desired level of performance should be the preferred network. Because with a larger degrees of freedom, the chances of having more local optima and getting stuck to one of them would usually be higher. Next, we investigate the effect of β on the performance of MRIT2NFS for a constant value of $f_{\rm th}$. A small value of β generates larger numbers of rules because of the smaller width of the initial type-2 fuzzy sets.

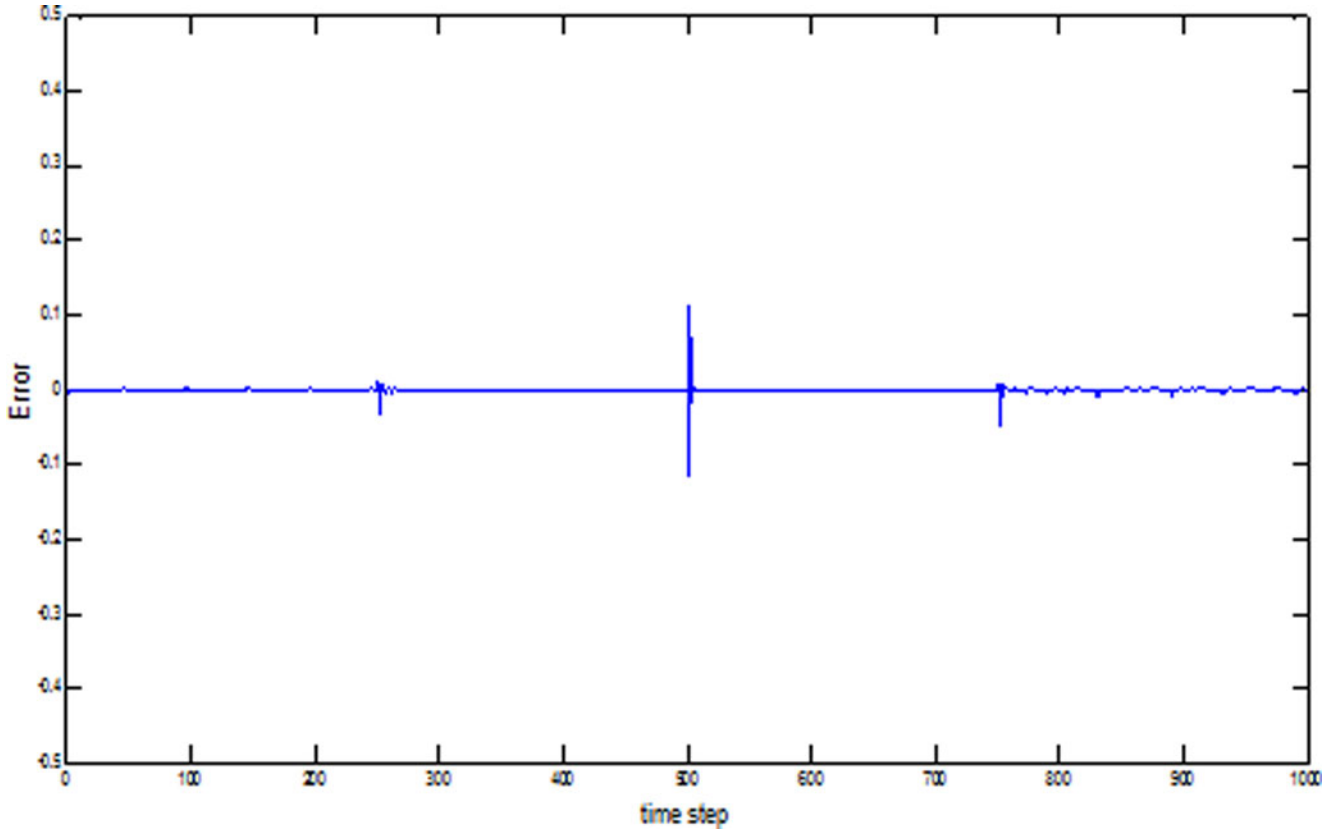

Fig. 4. Test errors between the MRIT2NFS and actual plant outputs.

TABLE III
INFLUENCE OF f_{th} AND ε ON THE PERFORMANCE OF MRIT2NFS WITH $\beta = 0.5$

Models	f_{th}	Number of Rules		Number of Parameters	Training RMSE	Test RMSE
MRIT2NFS	0.02	2		40	0.0008	0.0078
	0.15	3		63	0.0007	0.0075
	0.25	4		88	0.0009	0.0060
MRIT2NFS-ε	f_{th}	Number of Rules	ε	Number of Parameters	Training RMSE	Test RMSE
	0.02	2	0.4	40	0.0008	**0.0071**
			0.6	40	0.00077	0.0076
			0.8	38	0.0009	0.0088
	0.15	3	0.4	63	0.0005	**0.00772**
			0.6	61	0.00068	0.00768
			0.8	58	0.00075	0.0078
	0.25	4	0.4	87	0.0006	0.0055
			0.6	86	0.001	**0.0040**
			0.8	83	0.0007	0.0066

TABLE IV
INFLUENCE OF β ON THE PERFORMANCE OF MRIT2NFS WITH $f_{\rm th} = 0.02$

β	0.2	0.3	0.5	0.6	0.7
Number of Rules	5	4	2	2	2
Training RMSE	0.0016	0.0009	0.0008	**0.00076**	0.00078
Test RMSE	0.0098	0.0091	**0.0078**	0.008	0.0087

Table IV shows the performance of MRIT2NFS for different values of β when $f_{\rm th} = 0.02$. From Table IV, we observe that the network performance (both training and test error) is not sensitive to variations in β when $f_{\rm th} = 0.02$, although the number of rules decreases as β increases. It is interesting to note that as β increases from 0.2 to 0.7, the number of rules decreases from 5 to 2 without affecting the performance of the system. In fact, with two rules, both the training and test performances are slightly improved over the case with five rules. However, for a given problem, if the goal is to find the optimal parameters, we can use a two-level cross-validation mechanism.

Next, we assess how robust the network is with respect to measurement noise in the plant output. Since the plant output y_p is fed back as an input to the network, a noise in the measurement of the plant output y_p is likely to have an effect on the performance of the system. The experiment also uses the control input sequence in (50). We consider three levels of Gaussian noise with STDs 0.1, 0.3, and 0.5. We use 30 simulations for the statistical analysis. Table V shows the performance of MRIT2NFS for the three different noisy environments. For the purpose of comparison, we also use the same noisy environment to assess the noise tolerance of the other networks, including feedforward type-1 and type-2 FNN, TRFN [9], RSEIT2FNN [37], and RIFNN [38]. The results in Table V indicate that the feedforward type-2 FNN can deal with noise much better than the feedforward type-1 FNN. The consequent part of the TRFN is a function of current input variables. The consequent part in the RSEIT2FNN and the MRIT2NFS is of TSK-type that consists of system output y_p and its previous values. As a result of this, the rate of increase in the RMSE with increase in σ for MRIT2NFS may be marginally higher than that for other networks, which do not use feedback of system outputs. Finally, the MRIT2NFS achieves better performance than the other compared FNNs for noise-free and noisy cases.

B. Example 2 (Single-Input Single-Output Dynamic System Identification)

We now consider the following dynamic system with longer input delays:

$$y_p(t+1) = 0.72y_p(t) + 0.025y_p(t-1)u_1(t-1) + 0.01u_1^2(t-2) + 0.2u_1(t-3). \quad (51)$$

This plant is the same as the one used in [9]. This system has a single input ($n_u = 1$) and a single output ($n_o = 1$). Thus, the current values of $u(t)$ and $y_p(t)$ are fed as inputs to the MRIT2NFS input layer. The current output of the plant depends on one previous output and three previous inputs. Therefore, for MRIT2NFS, $N_1 = 3$ and $O_1 = 1$. The training data and time steps are the same as those used in Example 1. In MRIT2NFS training, the structure learning threshold is set to 0.02. After 90 epochs of training, two rules are generated. To test the identified system, the test signal used in Example 1 is also adopted here. Fig. 5 shows the outputs of the plant and those of the MRIT2NFS for these test inputs. Fig. 6 shows the test error between the outputs of MRIT2NFS and of the plant. Table VI shows the structure, and training and test RMSEs of MRIT2NFS. The performance of MRIT2NFS-ε ($\varepsilon = 0.6$ is used in this paper) with the same network size is also shown in Table VII. Like Example 1, Table VII shows that MRIT2NFS and MRIT2NFS-ε have similar performance. The performance of MRIT2NFS is compared with that of feedforward type-1 and typ-2 FNNs and recurrent type-1 FNNs. These models also use the same number of training epochs and training and test data as used for the MRIT2NFS. Table VI also depicts the number of rules and parameters, and the training and test RMSEs of these compared networks. These results show that the MRIT2NFS achieves better performance than that of other networks. In Table VII, we investigate the effect of different choices $f_{\rm th}$ and ε on the performance of MRIT2NFS when $\beta = 0.5$.

As done for Example 1, here also using the same computational protocols, we study the robustness of MRIT2NFS in noisy environments. Table VIII summarizes the results for the MRIT2NFS with different noise levels (Gaussian noise with STDs of 0.1, 0.3, and 0.5, and with 30 Monte Carlo realizations for each case). As revealed by Table VIII, for noisy environments, the MRIT2NFS achieves smaller RMSE than the other compared FNNs except the RIFNN. Note that, for noise-free data for the same problem, Table VI reveals that MRIT2NFS performs better than RIFNN, but with noisy data, RIFNN performs better. A possible reason for this may be that the consequent part in the RIFNN is a constant interval-valued set and is not a function of the system output, y_p, that are noisy. However, for MRIT2NFS, the rule consequents are functions of current output y_p, which are noisy. Therefore, the impact of noise on RIFNN is much weaker than that on MRIT2NFS. These results show that MRIT2NFS and MRIT2NFS-ε have similar performance. Table VIII reveals that the test error for the MRIT2NFS is smaller than that of the feedforward type-1 and type-2 FNNs and recurrent type-1 and type-2 FNNs.

C. Example 3 (Chaotic Series Prediction)

In this example, which is introduced in [41], we use MRIT2NFS to predict the chaotic behavior of a dynamic system with one delay and two sensitive parameters that are generated by the following equation:

$$y_p(t+1) = -P \cdot y_p^2(t) + Q \cdot y_p(t-1) + 1.0. \quad (52)$$

Equation (52), with $P = 1.4$ and $Q = 0.3$, produces a chaotic attractor. The system has no control input (i.e., $n_u = 0$) and a single output (i.e., $n_o = 1$) so that only output variable $y_p(t)$ is fed as input to the MRIT2NFS. It is a second-order system with one delay; therefore, $O_1 = 1$. The training procedure uses the plant output $y_p(t+1)$ as the desired output $y_d(t+1)$. Starting from the initial state $[y_p(1), y_p(0)] = [0.4, 0.4]$, 2000 patterns are

TABLE V
PERFORMANCE OF MRIT2NFS AND OTHER FEEDFORWARD AND RECURRENT MODELS WITH DIFFERENT NOISE LEVEL IN EXAMPLE 1

Models		Feedforward Type-1 FNN	TRFN-S [9]	Feedforward Type-2 FNN	RSEIT2FNN -UM[37]	RIFNN [38]	MRIT2NFS
Number of Rules		7	5	4	2	4	2
Number of Parameters		42	60	48	38	36	40
Test RMSE	STD=0.1	0.121	0.062	0.056	0.168	0.057	**0.021**
	STD=0.3	0.312	0.133	0.242	0.522	0.192	**0.05**
	STD=0.5	0.458	0.162	0.361	0.783	0.315	**0.098**

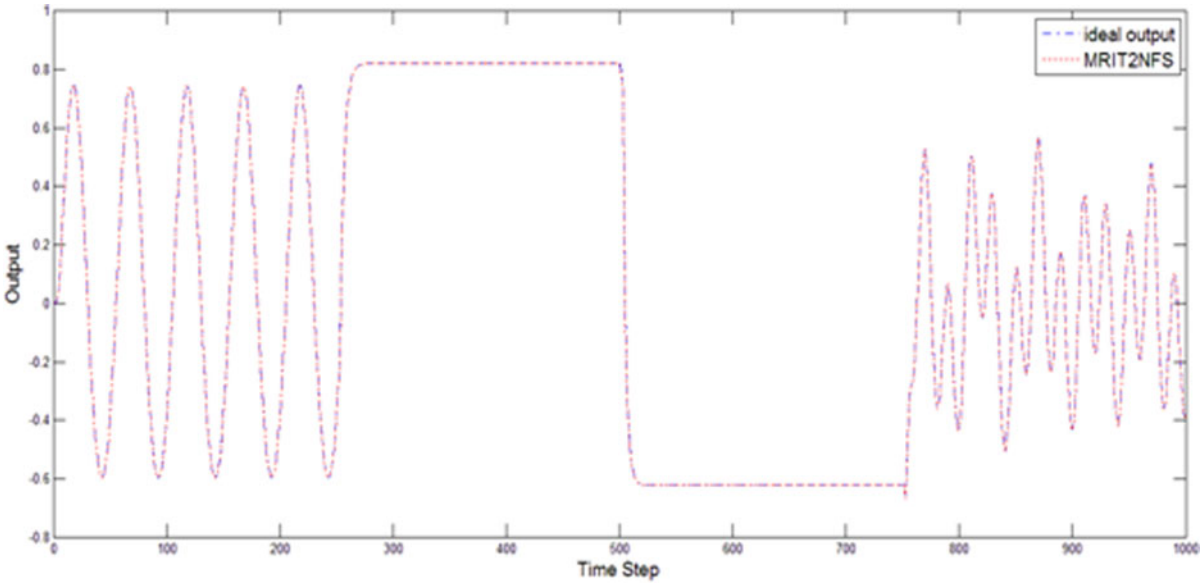

Fig. 5. Outputs of the dynamic plant (dashed-dotted line) and MRIT2NFS (dotted line) in Example 2.

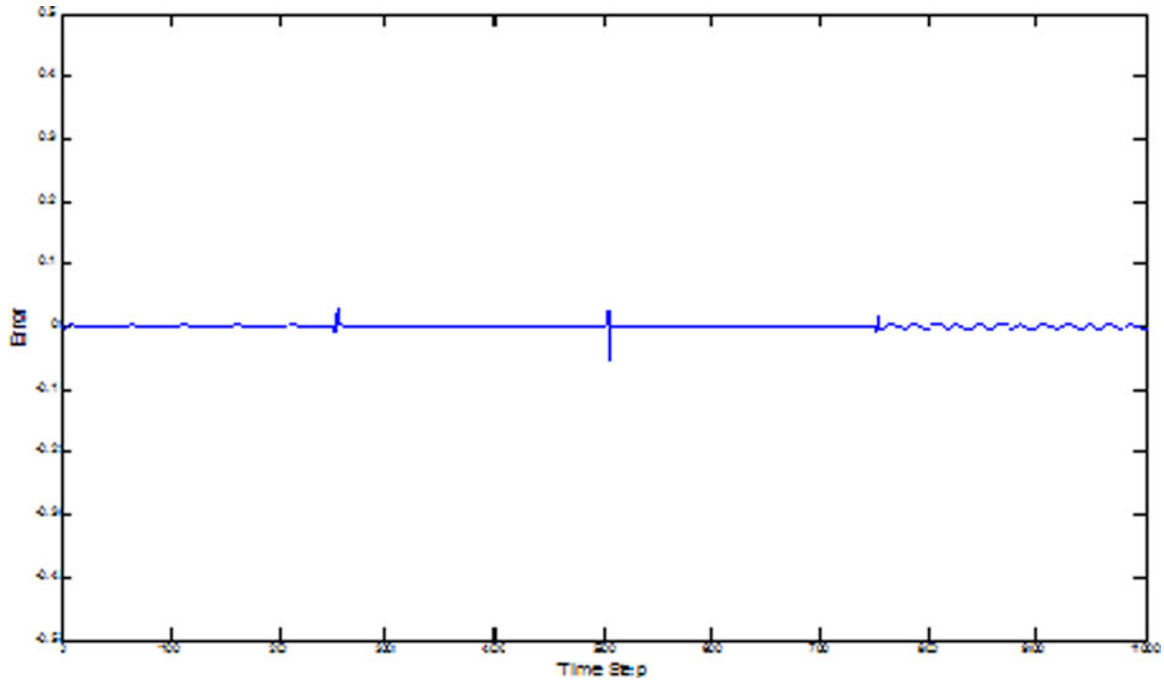

Fig. 6. Test errors between the MRIT2NFS and actual plant outputs.

TABLE VI
PERFORMANCE OF MRIT2NFS AND OTHER RECURRENT MODELS FOR SISO PLANT IDENTIFICATION IN EXAMPLE 2

Models	Feedforward Type-1 FNN	WRFNN [6]	TRFN-S [9]	RSEFNN [11]	Feedforward Type-2 FNN	RSEIT2FNN -UM[37]	RIFNN [38]	MRIT2NFS
Number of Rules	7	5	5	4	4	2	4	2
Number of Parameters	49	55	60	30	48	45	36	44
Training RMSE	0.007	0.0574	0.0076	0.0156	0.0155	0.0034	0.0125	**0.0013**
Test RMSE	0.032	0.083	0.0286	0.0279	0.038	0.006	0.0288	**0.0028**

TABLE VII
INFLUENCE OF $f_{\rm th}$ AND ε ON THE PERFORMANCE OF AN MRIT2NFS WITH $\beta = 0.5$

Models	f_{th}	Number of Rules		Number of Parameters	Training RMSE	Test RMSE
MRIT2NFS	0.02	2		44	0.0013	0.0028
	0.15	3		69	0.0005	0.00274
	0.25	4		96	0.0008	0.0032
MRIT2NFS-ε	f_{th}	Rules	ε	Number of Parameters	Training RMSE	Test RMSE
	0.02	2	0.4	44	0.00173	**0.0025**
			0.6	42	0.0011	0.0031
			0.8	42	0.0014	0.0032
	0.15	3	0.4	69	0.00051	**0.00287**
			0.6	69	0.00052	0.00366
			0.8	64	0.00059	0.0041
	0.25	4	0.4	96	0.0007	**0.0042**
			0.6	94	0.0065	0.0048
			0.8	87	0.0009	0.0051

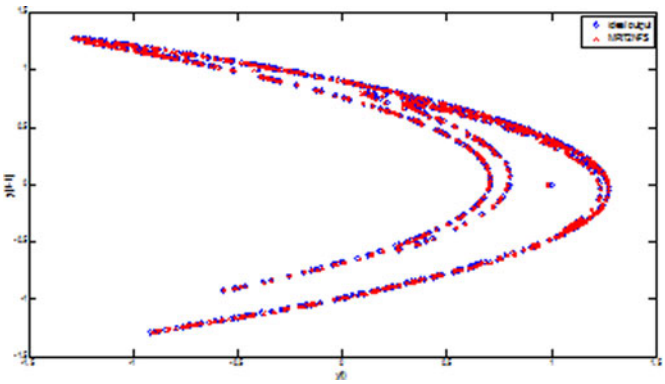

Fig. 7. Results of the phase plot for the chaotic system (O) and MRIT2NFS (X).

generated of which the first 1000 patterns are used for training and the remaining 1000 patterns are used for testing. In addition, the structure learning threshold $f_{\rm th}$ is set to 0.2, and the number of rules generated is 5 after 90 epochs of training.

Fig. 7 displays the phase plot of the actual and MRIT2NFS predicted results for the test patterns. Table IX includes the network size, and training and test RMSEs of MRIT2NFS. The performance of MRIT2NFS-ε is also depicted in Table IX. Like the other two examples, the performance of MRIT2NFS and MRIT2NFS-ε is quite similar. The performance of MRIT2NFS is also compared with that of feedforward type-1 and type-2 FNNs, and recurrent type-1 FNNs, including RFNN [5], WRFNN [6], TRFN-S [9], and RSEFNN [11]. From Table VIII, we find that MRIT2NFS-ε exhibits the best performance using almost the minimum number of free parameters, while MRIT2NFS achieves the next best performance, although it uses a few more free parameters.

In this case, we study the noise tolerance of our system with three levels of Gaussian noise with STDs of 0.3, 0.5, and 0.7. Based on 30 Monte Carlo realizations, in Table X, we summarize the test RMSEs of the feedforward type-1 and type-2 FNNs, recurrent type-1 FNNs, and MRIT2NFS. Table IX shows that the test error of the MRIT2NFS is much smaller than that of the RIFNN for noise-free data. However, Table X depicts that the test error of the MRIT2NFS for noisy environment is very close to that of the RIFNN. The possible reason for this is the same as explained in Example 2. These results also reveal that the test error of MRIT2NFS is smaller than those of the compared networks for all the three noise levels.

D. Example 4 (Multi-Input Multi-Output Dynamic System Identification)

In this example, we consider the plant described by

$$y_{p1}(t+1) = 0.5 \cdot \left[\frac{y_{p1}(t)}{1+y_{p2}^2(t)} + u_1(t-1) \right] \tag{53}$$

$$y_{p2}(t+1) = 0.5 \cdot \left[\frac{y_{p1}(t)y_{p2}(t)}{1+y_{p2}^2(t)} + u_2(t-1) \right]. \tag{54}$$

This MIMO dynamic system was also studied in [9]. This plant has two inputs ($n_u = 2$) and two outputs ($n_o = 2$). Therefore, four current input–output values $u_1(t), u_2(t), y_{p1}(t)$, and $y_{p2}(t)$ are fed to the input layer of the network. The present output of the plant depends on 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 MRIT2NFS are set to 1, 1, 0, and 0, respectively. The desired outputs for MRIT2NFS training are $y_{p1}(t+1)$ and $y_{p2}(t+1)$. The MRIT2NFS is trained in an online manner from time step $t = 1$ to $t = 11000$. The two control inputs $u_1(t)$ and $u_2(t)$ are independent and identically distributed uniform random sequences over $[-1.4, 1.4]$ for $t = 1$ to $t = 4000$. For the remaining 7000 time steps, sinusoid signals generated by $\sin(\pi t/45)$ are used for both $u_1(t)$ and $u_2(t)$. The learning coefficient η and the threshold $f_{\rm th}$ are set to 0.075 and 0.05, respectively. Based on a compromise between network size and performance, the threshold value 0.05 is used, as discussed in Example 1. For this dataset, the training results in three rules. Table XI shows the structure and RMSE of MRIT2NFS. To evaluate the effectiveness of the identified network, we use

TABLE VIII
PERFORMANCE OF MRIT2NFS AND OTHER FEEDFORWARD AND RECURRENT MODELS WITH DIFFERENT NOISE LEVEL IN EXAMPLE 2

Models		Feedforward Type-1 FNN	TRFN-S	Feedforward Type-2 FNN	RIFNN	RSEIT2FNN -UM	MRIT2NFS
Number of Rules		7	5	4	4	2	2
Number of Parameters		49	60	48	36	42	44
Test RMSE	STD=0.1	0.145	0.097	0.087	-	0.072	**0.056**
	STD=0.3	0.309	0.276	0.246	**0.15**	0.217	0.202
	STD=0.5	0.501	0.453	0.416	**0.23**	0.358	0.327

TABLE IX
PERFORMANCE OF MRIT2NFS AND OTHER FEEDFORWARD AND RECURRENT MODELS IN EXAMPLE 3

Models	Feedforward Type-1 FNN	WRFNN [6]	TRFN-S [9]	RSEFNN [11]	Feedforward Type-2 FNN	RSEIT2FNN -UM[37]	RIFNN [38]	MRIT2NFS -$\mathcal{E}$(0.6)	MRIT2NFS
Number of Rules	15	7	6	7	8	6	9	5	5
Number of Parameters	60	70	66	35	56	60	54	57	70
Training RMSE	0.234	0.191	0.0296	0.032	0.201	0.0043	0.073	**0.0028**	0.0041
Test RMSE	0.240	0.188	0.0245	0.021	0.200	0.0047	0.051	**0.0023**	0.0037

TABLE X
PERFORMANCE OF MRIT2NFS AND OTHER FEEDFORWARD AND RECURRENT MODELS WITH DIFFERENT NOISE LEVEL IN EXAMPLE 3

Models		Feedforward Type-1 FNN	TRFN-S	Feedforward Type-2 FNN	RIFNN	RSEIT2FNN -UM	MRIT2NFS
Number of Rules		15	5	4	9	6	5
Number of Parameters		60	60	48	54	60	70
Test RMSE	STD=0.3	0.621	0.604	0.617	0.575	0.528	**0.513**
	STD=0.5	1.114	0.955	0.917	**0.725**	0.803	0.814
	STD=0.7	1.706	1.256	1.180	1.040	**1.020**	1.032

the following two control input sequences:

$$u_1(t) = u_2(t) = \begin{cases} \sin(\pi t/25), & 1001 \le t < 1250 \\ 1.0, & 1250 \le t < 1500 \\ -1.0, & 1500 \le t < 1750 \\ 0.3\sin(\frac{\pi t}{25}) + 0.1\sin(\frac{\pi t}{32}) + 0.6\sin(\frac{\pi t}{10}), & 1750 \le t < 2000. \end{cases} \tag{55}$$

Fig. 8 shows a very good match between the actual output and the network output. Table XI shows the test RMSEs of y_{p1} and y_{p2}. We also compare the performance of MRIT2NFS with that of memory NN (MNN) [39], feedforward type-1 and type-2 FNNs, and recurrent type-1 FNNs. The MNN is a kind of recurrent NN and has been applied to the same problem in [9]. For the recurrent FNNs, we use the same training data, test data, and the number of training epochs as those for MRIT2NFS, except in the case of the MNN, where a total number of 77 000 time steps are used in [39]. Table XI shows that the performance of MRIT2NFS is better than that of feedforward and recurrent networks.

In this case also, we investigate the noise tolerance of MRIT2NFS with the same three levels of noise that are used in the previous example. Table XII summarizes the results for feedforward type-1 and type-2 FNNs, TRFN [9], RSEIT2FNN [36], and MRIT2NFS over 30 Monte Carlo realizations. In this case too, we find that under noisy environments, the test RMSEs of MRIT2NFS is better than those of the compared networks.

TABLE XI
PERFORMANCE OF MRIT2NFS AND OTHER RECURRENT MODELS FOR MIMO PLANT IDENTIFICATION IN EXAMPLE 4

Models	Feedforward Type-1 FNN	MNN [29]	RFNN [5]	WRFNN [6]	TRFN-S [9]	Feedforward Type-2 FNN	RSEIT2FNN [37]	MRIT2NFS
Number of Rules	7	-	13	7	7	4	3	3
Number of Parameters	126	131	182	182	189	128	126	132
Training RMSE	0.0422	-	0.0687	0.0449	0.0382	0.0496	**0.0036**	0.0039
Test y_{p1} RMSE	0.0373	0.0186	0.0824	0.0472	0.0396	0.0411	0.0081	**0.0066**
Test y_{p2} RMSE	0.0480	0.0327	0.0801	0.0502	0.0383	0.0423	**0.0113**	0.0116

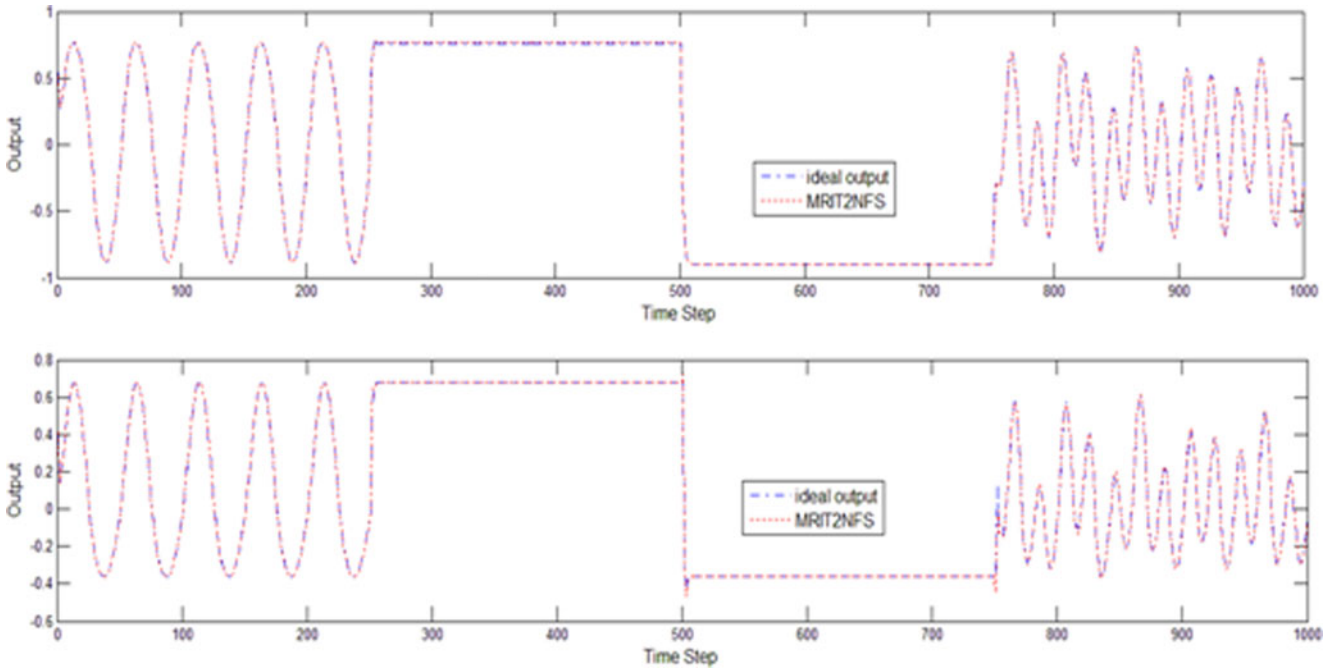

Fig. 8. Output of the MIMO system (dashed-dotted curve) and MRIT2NFS (dotted curve) in Example 4. (a) Output y_{p1}. (b) Output y_{p2}.

TABLE XII
PERFORMANCE OF MRIT2NFS AND OTHER MODELS WITH DIFFERENT NOISE LEVELS IN EXAMPLE 4

Networks		Feedforward Type-1 FNN	Feedforward Type-2 FNN	TRFN-S	RSEIT2FNN	MRIT2NFS
Number of Rules		7	4	7	3	3
Number of Parameters		126	128	189	126	132
Test RMSE y_{p1}	STD=0.3	0.256	0.189	0.188	**0.16**	0.165
	STD=0.5	0.417	0.307	0.316	0.316	**0.258**
	STD=0.7	0.578	0.420	0.427	0.427	**0.354**
Test RMSE y_{p2}	STD=0.3	0.197	0.145	0.143	0.090	**0.078**
	STD=0.5	0.322	0.233	0.226	0.155	**0.122**
	STD=0.7	0.443	0.323	0.298	0.227	**0.193**

TABLE XIII
PERFORMANCE OF MRIT2NFS-ε AND OTHER MODELS FOR MODELING OF NONLINEAR SYSTEM IN EXAMPLE 5

Models	Feedforward Type-1 FNN	TRFN-S [9]	Feed-forward Type-2 FNN	MRIT2NFS -$\mathcal{E}$ (0.8)
Iteration	100	100	100	100
Number of Rules	5	3	3	3
Number of Parameters	35	30	36	40
Test RMSE	0.312	0.244	0.26	**0.206**

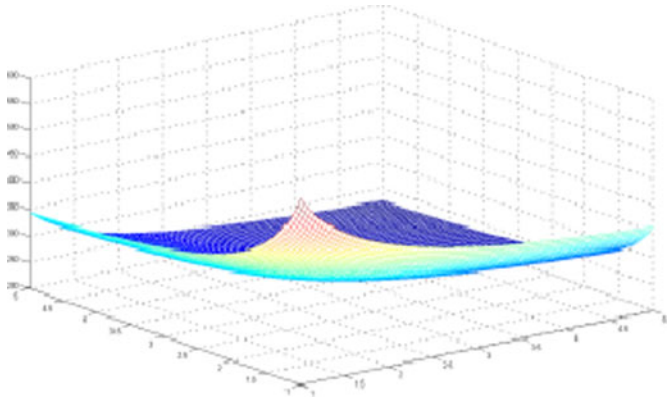

Fig. 9. Pictorial representation of HANG.

E. Example 5 (HANG Nonlinear System)

This example uses an MRIT2NFS-ε to deal with the nonlinear characteristics of a plant. This is a very well-studied system, but it is not recurrent in nature. The HANG [42] system is defined by the equation:

$$y = (1 + x_1^{-1} + x_2^{-1.5})^2, \quad 1 \le x_1, x_2 \le 5. \tag{56}$$

To obtain the training data as done in [42], we have generated 50 random pairs of (x_1, x_2),$1 \le x_1, x_2 \le 5$, and computed the corresponding outputs using (56). This system has only two current control inputs $x_1(t)$ and $x_2(t)$ ($n_u = 2$) and no external output ($n_o = 0$). Therefore, only two input states x_1 and x_2 are fed as input to the MRIT2NFS. The current output of the plant depends on current control inputs with no time delay. Thus, in this case, we set $N_1 = 0$ and $O_1 = 0$ in the MRIT2NFS consequent part. Fig. 9 shows a pictorial representation of HANG. The threshold f_{th} is set to be 0.02, and the number of rules is 3 after 100 epochs of training. Table XIII shows the number of rules, parameters, and test RMSE of the MRIT2NFS. For comparison, Table XIII also shows the test RMSEs of feedforward type-1 and type-2 FNNs, and TRFN using the same I–O data. The result shows that the test error of the MRIT2NFS-ε is marginally better than feedforward type-2 FNN. Since this is not a dynamic system, the recurrent architecture is not likely to yield any additional benefits.

V. CONCLUSION

In this paper, we have proposed a novel recurrent type-2 neural fuzzy system called MRIT2NFS to identify time-varying systems (systems with temporal behavior). Identification of such systems is difficult because the plant output depends on the present state as well as on previous states and past outputs. The proposed MRIT2NFS approach is quite effective in modeling dynamic systems. It is equally effective indentifying nondynamical (algebraic) systems also. For MRIT2NFS learning, we have used the type-2 fuzzy set theoretic concepts to evolve the structure of the network in an online manner. Our online structure learning algorithm enables the network to efficiently identify the required structure of the network—we do not need to set any initial MRIT2NFS structure in advance. We use the rule-ordered Kalman filter algorithm to tune the consequent parameters to yield a very effective learning. We have also proposed a strategy to eliminate redundant recurrent weights. This is quite effective particularly when we have more rules. We have tested our system on several dynamic systems and one algebraic (nonrecurrent) system and compared the performance with several existing state-of-the-art systems. We have compared the performance of our system with both type-1 and type-2 state-of-the-art FNNs. Among the two type-1 FNNs, one is of the feedforward type and the other is a recurrent network with linear consequents. Our results have demonstrated the consistently superior performance of MRIT2NFS over both the type-1 (recurrent and feedforward) systems as well as recurrent type-2 systems. We have demonstrated the superior performance both in terms of modeling capability as well as noise handling capability. Our system is found to be quite robust with respect to noisy data. In order to make a fair comparison, we have tried to keep the number of free parameters in all systems comparable.

In this investigation, we have assumed knowledge about the system order and number of delayed inputs. In the absence of this information, one can use sufficiently large number of delayed inputs and past outputs in the consequents and then find the useful ones using a concept similar to the feature attenuating gates as done in [44]. However, use of such a concept may not ensure that all useful delayed inputs and outputs are consecutive in time. The other alternative could be to use a cross-validation scheme to find the best combination of number of lagged inputs and outputs to be used in the consequents. In the future, we would like to study such possibilities and also make theoretical study on the convergence of MRIT2NFS learning algorithm.

REFERENCES

[1] J. B. Theocharis and G. Vachtsevanos, "Recursive learning algorithms for training fuzzy recurrent models," *Int. J. Intell. Syst.*, vol. 11, no. 12, pp. 1059–1098, 1996.

[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] 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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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, pp. 2552–2568, Oct. 2010.

[12] 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.

[13] 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.

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

[15] 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.

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

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] C. H. Wang 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.

[24] 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.

[25] 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. Robotics Autom.*, Jul. 16–20, 2003, vol. 3, pp. 1264–1269.

[26] 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.

[27] 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.

[28] Y. C. Lin and C. H. Lee, "System identification and adaptive filter using a novel fuzzy neuro system," *Int. J. Comput. Cognition*, vol. 5, no. 1, pp. 15–26, 2007.

[29] W. S. Chan, C. Y. Lee, C. W. Chang, and Y. H. Chang, "Interval type-2 fuzzy neural network for ball and beam systems," in *Proc. IEEE conf. Syst. Sci. Eng.*, Jul. 1–3, 2010, pp. 315–320.

[30] C. F. Juang and Y. W. Tsao, "A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 1411–1424, Dec. 2008.

[31] C. D. Li, J. Q. Yi, and D. B. Zhao, "Interval type-2 fuzzy neural network controller (IT2FNNC) and its application to a coupled-tank liquid-level control system," in *Proc. Int. Conf. Innovative Comput. Inf. Control*, Jun. 18–20 2008, p. 508.

[32] F. J. Lin and P. H. Chou, "Adaptive control of two-axis motion control system using interval type-2 fuzzy neural network," *IEEE Trans. Ind. Electron.*, vol. 56, no. 1, pp. 178–193, Jan. 2009.

[33] C. F. Juang, R. B. Huang, and W. Y. Cheng, "An interval type-2 fuzzy neural network with support vector regression for noisy regression problems," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 4, pp. 686–699, Aug. 2010.

[34] R. H. Abiyev and O. Kaynak, "Type-2 fuzzy neural structure for identification and control of time-varying plants," *IEEE Trans. Ind. Electron.*, vol. 57, no. 12, pp. 4147–4159, Dec. 2010.

[35] C. H. Lee, H. H. Chang, C. T. Kuo, J. C. Chien, and T. W. Hu, "A novel recurrent interval type-2 fuzzy neural network for nonlinear channel equalization," in *Proc. Int. Muticonf. Eng. Computer Scientists*, Mar. 18–20, 2009, vol. 1, pp. 1231–1240.

[36] Y. Y. Lin, J. Y. Chang, and C. T. Lin, "An internal/interconnection recurrent type-2 fuzzy neural network (IRT2FNN) for dynamic system identification," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct 10–13, 2010, pp. 733–737.

[37] 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.

[38] C. F. Juang, Y. Y. Lin, and R. B. Huang, "Dynamic system modeling using a recurrent interval-valued fuzzy neural network and its hardware implementation," *Fuzzy Sets Syst.*, vol. 179, pp. 83–99, May 2011.

[39] 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.

[40] 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.

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

[42] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 1, pp. 7–31, Feb. 1993.

[43] A. Laha, N. R. Pal, and J. Das, "Land cover classification using fuzzy rules and aggregation of contextual information through evidence theory," *IEEE Trans. Geosci. Remote Sensing*, vol. 24, no. 6, pp. 1633–1641, Jun. 2006.

[44] N. R. Pal and S. Saha, "Simultaneous structure identification and fuzzy rule generation for Takagi–Sugeno models," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 6, pp. 1626–1638, Dec. 2008.

[45] C. S. Lee, M. H. Wang, and H. Hagras, "A type-2 fuzzy ontology and its application to personal diabetic-diet recommendation," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 2, pp. 374–395, Apr. 2010.

[46] D. Hidalgo, O. Castillo, and P. Melin, "Type-1 and Type-2 fuzzy inference systems as integration methods in modular neural networks for multimodal biometry and its optimization with genetic algorithms," *Inf. Sci.*, vol. 179, pp. 2121–2145, 2009.

[47] X. Chen, Y. Li, R. Harrison, and Y. Q. Zhang, "Type-2 fuzzy logic-based classifier fusion for support vector machines," *Appl. Soft Comput.*, vol. 8, pp. 1222–1231, 2008.

[48] X. Liu and J. M. Mendel, "Connect Karnik–Mendel algorithms to root-finding for computing the centroid of an interval type-2 fuzzy set," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 4, pp. 652–565, Aug. 2011.

[49] D. Wu, J. M. Mendel, and S. Coupland, "Enhanced interval approach for encoding words into interval type-2 fuzzy sets and its convergence analysis," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 3, pp. 499–513, Jun. 2012.

[50] D. Wu and J. M. Mendel, "Linguistic summarization using IF–THEN rules and interval type-2 fuzzy sets," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 1, pp. 136–151, Feb. 2011.

[51] O. Linda and M. Manic, "General type-2 fuzzy c-means algorithm for uncertain fuzzy clustering," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 5, pp. 883–897, Oct. 2012.
[52] D. Wu and J. M. Mendel, "On the continuity of type-1 and interval type-2 fuzzy logic systems," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 1, pp. 179–192, Feb. 2011.
[53] C. Y. Yeh, W. H. R. Jeng, and S. J. Lee, "An enhanced type-reduction algorithm for type-2 fuzzy sets," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 2, pp. 227–240, Apr. 2011.
[54] D. Zhai and J. M. Mendel, "Enhanced centroid-flow algorithm for computing the centroid of general type-2 fuzzy sets," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 5, pp. 939–956, Oct. 2012.
[55] D. Zhai and J. M. Mendel, "Comment on "toward general type-2 fuzzy logic systems based on zSlices," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 5, pp. 996–997, Oct. 2012.
[56] D. Zhai and J. M. Mendel, "Computing the centroid of a general type-2 fuzzy set by means of the centroid-flow algorithm," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 3, pp. 401–422, Jun. 2011.
[57] O. Linda and M. Manic, "Monotone centroid flow algorithm for type reduction of general type-2 fuzzy sets," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 5, pp. 805–819, Oct. 2012.
[58] R. Hosseini, S. D. Qanadli, S. Barman, M. Mazinani, T. Ellis, and J. Dehmeshki, "An automatic approach for learning and tuning Gaussian interval type-2 fuzzy membership functions applied to lung CAD classification system," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 2, pp. 224–234, Apr. 2012.
[59] M. Nie and W. W. Tan, "Analytical structure and characteristics of symmetric Karnik–Mendel type-reduced interval type-2 fuzzy PI and PD controllers," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 3, pp. 416–430, Jun. 2012.
[60] D. Wu, "On the fundamental differences between interval type-2 and type-1 fuzzy logic controllers," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 5, pp. 832–848, Oct. 2012.
[61] S. Barkat, A. Tlemcani, and H. Nouri, "Noninteracting adaptive control of PMSM using interval type-2 fuzzy logic systems," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 5, pp. 925–936, Oct. 2011.

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 working toward 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 networks, and field-programmable gate array 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 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, NC, USA, in 1987.

During 1976–1978 and 1980–1982, he was a Research Fellow with the Chung Shan Institute of Science and Technology, Lung-Tan, Taiwan. 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.

Nikhil R. Pal (F'05) is currently a Professor with the Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata, India. His current research interest includes bioinformatics, brain science, fuzzy logic, image and pattern analysis, neural networks, and evolutionary computation.

Dr. Pal was the Editor-in-Chief of the IEEE TRANSACTIONS ON FUZZY SYSTEMS from January 2005 to December 2010. He has served/been serving on the Editorial/Advisory Board/Steering Committee of several journals, including the *International Journal of Approximate Reasoning, Applied Soft Computing, Neural Information Processing—Letters and Reviews*, the *International Journal of Knowledge-Based Intelligent Engineering Systems*, the *International Journal of Neural Systems, Fuzzy Sets and Systems*, the *International Journal of Intelligent Computing in Medical Sciences and Image Processing, Fuzzy Information and Engineering: An International Journal*, the IEEE TRANSACTIONS ON FUZZY SYSTEMS, and the IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS—PART B: CYBERNETICS. He has given many plenary/keynote speeches in different premier international conferences in the area of computational intelligence. He has served as the General Chair, Program Chair, and Co-Program Chair of several conferences. He was a Distinguished Lecturer of the IEEE Computational Intelligence Society (CIS) and was a member of the Administrative Committee of the IEEE CIS (2010–2012). At present, he is the Vice President for Publications of the IEEE CIS. He is the General Chair of the 2013 IEEE International Conference on Fuzzy Systems. He is a Fellow of the National Academy of Sciences, India, a Fellow of the Indian National Academy of Engineering, a Fellow of the Indian National Science Academy, and a Fellow of the International Fuzzy Systems Association.

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, USA, in 1989 and 1992, respectively.

He is currently the Provost, Chair Professor of Electrical and Computer Engineering, and Director of the Brain Research Center, National Chiao Tung University. He has published more than 120 journal papers in the areas of neural networks, fuzzy systems, multimedia hardware/software, and cognitive neuroengineering, including approximately 74 IEEE journal papers.

Dr. Lin became a Fellow of the IEEE for his contributions to biologically inspired information systems in 2005. He is the Editor-in-Chief of the IEEE TRANSACTIONS ON FUZZY SYSTEMS. He also served on the Board of Governors of the IEEE Circuits and Systems Society during 2005–2008, the IEEE Systems, Man, and Cybernetics Society during 2003–2005, the IEEE Computational Intelligence Society during 2008–2010, and Chair of IEEE Taipei Section during 2009–2010. He served as the Deputy Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS during 2006–2008. He is the co-author of *Neural Fuzzy Systems* (Englewood Cliffs, NJ: Prentice-Hall) and the author of *Neural Fuzzy Control Systems with Structure and Parameter Learning* (Singapore: World Scientific).