A fuzzy-neural multi-model for nonlinear systems identification and control

Ieroham S. Baruch*, Rafael Beltran Lopez, Jose-Luis Olivares Guzman, Jose Martin Flores

CINVESTAV-IPN, Department of Automatic Control, Av. IPN No. 2508, A.P. 14-740, Col. Zacatenco, 07360 Mexico D.F., Mexico

Received 10 March 2006; received in revised form 18 May 2007; accepted 18 January 2008
Available online 12 February 2008

Abstract

The paper proposed to apply a hierarchical fuzzy-neural multi-model and Takagi–Sugeno (T–S) rules with recurrent neural procedural consequent part for systems identification, states estimation and adaptive control of complex nonlinear plants. The parameters and states of the local recurrent neural network models are used for a local direct and indirect adaptive trajectory tracking control systems design. The designed local control laws are coordinated by a fuzzy rule-based control system. The upper level defuzzification is performed by a recurrent neural network. The applicability of the proposed intelligent control system is confirmed by simulation examples and by a DC-motor identification and control experimental results. Two main cases of a reference and plant output fuzzification are considered—a two membership functions without overlapping and a three membership functions with overlapping. In both cases a good convergent results are obtained.

Keywords: Recurrent neural networks; Fuzzy-neural hierarchical multi-model; Systems identification; Indirect adaptive control; Direct adaptive control; DC-motor control

1. Introduction

In the last decade, the computational intelligence (CI), including artificial neural networks (ANN) and fuzzy systems (FS) became a universal tool for many applications. Because of their approximation and learning capabilities [14], the ANNs have been widely employed to dynamic process modelling, identification, prediction and control [15,24,27–29,34]. Mainly, two types of ANN models are used: feedforward (FFNN) or static and recurrent (RNN) or dynamic. The first type of ANN could be used to resolve dynamic tasks introducing external dynamic feedbacks. The second one possesses its own internal dynamics performed by its internal local feedbacks so to form memory neurons [11,29]. The application of the FFNN for modelling, identification and control of nonlinear dynamic plants caused some problems which could be summarised as follows: (1) The dynamic systems modelling usually is based on the nonlinear autoregressive moving average (NARMA) model which need some information of input/output model...
orders, and input and output tap-delays ought to be used, [14]. (2) The FFNN application for multi-input multi-output systems identification needs some relative order structural information. (3) The ANN model structure ought to correspond to the structure of the identified plant where four different input/output plant models are used [24]. (4) The lack of universality in ANN architectures caused some difficulties in its learning and a Backpropagation (BP) through time learning algorithm needs to be used [14]. (5) Most of NARMA-based ANN models are sequential in nature and introduced a relative plant-dependent time-delay. (6) Most of the ANN-based models are nonparametric ones [24], and so, not applicable for an indirect adaptive control systems design. (7) All this ANN-based models do not perform state and parameter estimation at the same time [15,28]. (8) All these models are appropriate only for identification of nonlinear plants with smooth, single, odd, nonsingular nonlinearities.

The major disadvantage of all these approaches is that the identification ANN model applied is a nonparametric one that does not permit them to use the obtained information directly for control systems design objectives. In [5,6,9], Baruch and co-authors applied the state-space approach to describe RNN in an universal way, defining a Jordan canonical two- or three-layer RNN model, named recurrent trainable neural network (RTNN) which has a minimum number of learned parameter weights. This RNN model is a parametric one, permitting to use of the obtained parameters during the learning for control systems design. Furthermore, the RTNN model is a system state predictor/estimator, which permits to use the obtained system states directly for state-space control. This model has the advantage to be completely parallel one, so its dynamics depends only on the previous step and not on the other past steps, determined by the systems order which simplifies the computational complexity of the learning algorithm with respect to the sequential RNN model of Frasconi, Gori and Soda (FGS-RNN) [11].

Similarly to the static ANNs, the fuzzy models could approximate static nonlinear plants where structural plant information is needed to extract the fuzzy rules [3,21]. The difference between them is that the ANN model are global models where training is performed on the entire pattern range and the FS models perform a fuzzy blending of local models space based on the partition of the input space. So the aim of the neuro-fuzzy (fuzzy-neural) model is to merge both ANN and FS approaches so to obtain fast adaptive models possessing learning [19]. The fuzzy-neural networks are capable of incorporating both numerical data (quantitative information) and expert’s knowledge (qualitative information), and describe them in the form of linguistic IF–THEN rules. During the last decade considerable research has been devoted towards developing recurrent neuro-fuzzy models. The first attempt was made in [16] where an ANFIS with external feedback is used as a neuro-fuzzy controller. Through BP learning, ANFIS is adapted to refine, or derive the fuzzy IF–THEN rules using system input–output data. Due to the recurrent neuro-fuzzy model with internal dynamics, the recurrent fuzzy rules introduced the feedback in the antecedent and the consequent part of the model [1,12,17,32,35], which is in fact a computational procedure. To resolve dynamic problems in [19,20] a compensation-based recurrent fuzzy neural network (CRFNN) is proposed. This is a recurrent multi-layer connectionist network for fuzzy reasoning constructed from a set of different fuzzy rules (normal, compensatory, recurrent) learned by different BP learning laws. The main disadvantage here is the lack of universality where different layers with different concepts, different membership functions and different BP learning laws should be applied. Furthermore, the CRFNN require using a lot of fuzzy rules so to perform a good approximation. The error cost function used for a BP learning of the Gaussian membership function parameters in some cases is not convex which makes the learning convergence very time consuming [14]. To reduce the number of IF–THEN rules, the hierarchical approach could be used [25]. A promising approach of recurrent neuro-fuzzy systems with internal dynamics is the application of the Takagi–Sugeno (T–S) fuzzy rules with a static premise and a dynamic function consequent part [22,30,31]. The [4,7,10,22,23] proposed as a dynamic function in the consequent part of the T–S rules to use an RNN. Some extension of this T–S approach for identification of dynamic plants with distributed parameters is given in [26]. The difference between the approach used in [22,23] fuzzy neural model and the approach used in [7,10,4] is that the first one uses the FGS-RNN [11] model, which is sequential one, and the second one uses the RTNN model [4], which is completely parallel one. But it is not still enough because the neural nonlinear dynamic function ought to be learned, and the BP learning algorithm is not introduced in the T–S fuzzy rule.

So, the present paper proposed to extend the power of the fuzzy rules, using in its consequent part a learning procedure instead of dynamic nonlinear function and to organize the defuzzification part as a second RNN hierarchical level incorporated in a new hierarchical fuzzy-neural multi-model (HFNNM) architecture. The output of the upper level represents a filtered weighted sum of the outputs of the lower level RTNN models. The fuzzy-neural hierarchical multi-model proposed uses only three membership functions (positive, zero and negative), which combine the advantages of the RNNs with that of the fuzzy logic, simplifying the structure, augmenting the level of adaptation and decreasing
the noise. Also we propose in the consequent part of the fuzzy rules to use a learning procedures or sequences of procedures which extend its computational capabilities incorporating identification and control features in the same architecture.

2. RTNN model and control laws description

2.1. Architecture and learning of the RTNN

The RTNN model is described by the following equations [4,7,9,10]:

\[
X(k + 1) = JX(k) + BU(k); \quad J = \text{block-diag}(J_i); \quad |J_i| < 1, \quad i = 1, \ldots, N,
\]

\[
Z(k) = \Gamma[X(k)],
\]

\[
Y(k) = \Phi[CZ(k)],
\]

where \( X(k) \) is an \( N \)-state vector, \( U(k) \) is a \( M \)-input vector, \( Y(k) \) is an \( L \)-output vector, \( Z(k) \) is an \( L \)-vector—output of the hidden layer, \( \Gamma(\cdot), \Phi(\cdot) \) are vector-valued activation functions like saturation, sigmoid or hyperbolic tangent, which have compatible dimensions, \( J \) is a weight-state diagonal matrix with elements \( J_i \) and the inequality in (1) is a stability preserving condition, imposed on the weights \( J_i, B \) and \( C \) are weight input and output matrices with compatible dimensions and block structure, corresponding to the block structure of \( J \). The given RTNN model is a completely parallel parametric one, with parameters—the weight matrices \( J, B, C \) and the state vector \( X(k) \).

The RTNN topology has a linear time varying structure properties like controllability, observability, reachability and identifiability, which are considered in [5]. These properties of the RTNN structure signify that starting from the block-diagonal matrix structure of \( J \), we can find a correspondence in the block structure of the matrices \( B \) and \( C \), that is show us how to find out the ability of learning of this RTNN. The main advantage of this discrete RTNN (which is really a Jordan canonical RNN model), is of being an universal hybrid neural network model with one or two feedforward layers, and one recurrent hidden layer, where the weight matrix \( J \) is a block-diagonal one. So, the RTNN posses a minimal number of learning weights and the performance of the RTNN is fully parallel. The described RTNN architecture could be used as one step ahead state predictor/estimator and systems identifier. Another property of the RTNN model is that it is globally nonlinear, but locally linear. That is why the matrices \( J, B, C \), generated by learning, could be used to design a linear sliding mode control law [8,10]. Furthermore, the RTNN model is robust, due to the dynamic weight adaptation law, based on the sensitivity model of the RTNN, and the performance index (written in two forms), which is as follows:

\[
\zeta(k) = (1/2) \sum_j [E_j(k)]^2, \quad j \in C; \quad \zeta = (1/N_e) \sum_k \zeta(k).
\]

Here the performance index \( \zeta(\cdot) \) is an instantaneous mean squared error (MSE) and it is a nonlinear function of the weight matrices of the output and the hidden RTNN layers, respectively; the performance index \( \zeta \) is the total MSE for one epoch of learning with dimension \( N_e \). The general RTNN-BP learning algorithm, written in vector-matricial form, is given by the following equation:

\[
W(k + 1) = W(k) + \eta \Delta W(k) + \alpha \Delta W(k - 1); \quad |W_{ij}(k + 1)| < W_0,
\]

where \( W(k) \) is the weight matrix, being modified \((J(k), B(k), C(k))\), \( \Delta W(k) \) is the weight matrix correction \((\Delta J(k), \Delta B(k), \Delta C(k))\), \( \eta \) is a learning rate normalized parameter’s diagonal matrix, and \( \alpha \) is a momentum term normalized learning parameter’s diagonal matrix. The inequality in (5) represents an anti-windup condition, where the learned weight \( W_{ij}(k) \) is restricted in a specified region \( W_0 \). The momentum term of this learning algorithm is used when some error oscillations occurred. The general BP learning algorithm (5) could be applied on-line, in real time, where the instantaneous MSE \( \zeta(\cdot) \) is minimized with respect to the RTNN weight matrices each learning iteration, or off-line, where the total MSE \( \zeta \) is minimized once for epoch of learning \( N_e \). The weight matrix elements update for the discrete time model of the RTNN has been given and applied in [4], but here it is expressed in vector-matricial form [9], using the RTNN adjoint derived applying the diagrammatic method, proposed by Wan and Beaufays [33]. The weight update algorithm is like that:

\[
\Delta C(k) = E_1(k)Z^T(k); \quad E_1(k) = \Phi'[Y(k)]E(k); \quad E(k) = Y_\delta(k) - Y(k),
\]
\[ \Delta J(k) = E_3(k)X^T(k); \quad E_3(k) = J'(Z(k))E_2(k); \quad E_2(k) = C^T(k)E_1(k), \]  
\[ \Delta v J(k) = E_3(k) \otimes X(k), \]  
\[ \Delta B(k) = E_3(k)U^T(k), \]

where the super index \( T \) means vector transpose, \( \Delta J(k), \Delta B(k), \Delta C(k) \), are weight corrections of the of the learned matrices \( J(k), B(k), C(k) \), respectively, \( E(k) = Y_d(k) - Y(k) \) is an \( L \)-error vector of the output RTNN layer, where \( Y_d(k) \) is a desired target vector (plant output for identification tasks) and \( Y(k) \) is a RTNN output vector, both with dimensions \( L \), \( X(k) \) is an \( N \)-state vector, and \( E_j(k) \) is a \( j \)th error vector with respective dimension, \( I'(\cdot), \Phi'(\cdot) \) are diagonal Jacobian matrices with appropriate dimensions, which elements are derivatives of the respective activation functions. Eq. (7) represented the learning of the feedback weight matrix of the hidden layer, which is supposed as a full \( (N \times N) \) matrix. Eq. (8) gives the learning solution when this matrix is diagonal, where \( v J(k) \) is an \( N \)-vector, which is the diagonal of the matrix \( J(k) \). The Eqs. (1)–(3) together with Eqs. (6)–(9) forms a BP-learning procedure, where the functional algorithm (1)–(3) represented the forward step, executed with constant weights, and the learning algorithm (6)–(9) represented the backward step, executed with constant signal vector variables. This learning procedure, denoted by \( \Pi (L, M, N, Y_d, U, X, J, B, C, E) \), could be executed on-line or off-line. It uses as input data the RTNN model dimensions \( L, M, N \), and the learning data vectors \( Y_d(k), U(k) \), and as output data—the \( X(k) \)-state vector, and the matrix weight parameters \( J(k), B(k), C(k) \).

2.2. Stability proofs of the learning algorithm

The stability and the properties of the BP-RTNN learning algorithm, given by Eq. (5), are proved by one theorem and two lemmas.

*Theorem of stability*: Let the RTNN with Jordan canonical structure is given by Eqs. (1)–(3) and the nonlinear plant model [5] is as follows:

\[ X_d(k + 1) = F[X_d(k), U(k)], \]
\[ Y_d(k) = G[X_d(k)], \]

where \( Y_d(k), X_d(k), U(k) \) are output, state and input variables with dimensions \( L, N_d, M \), respectively; \( F(\cdot), G(\cdot) \) are vector valued nonlinear functions with respective dimensions. Under the assumption of RTNN identifiability made, the application of the BP learning algorithm for \( J(k), B(k), C(k) \) in general matricial form, described by Eq. (5), and the learning rates \( \eta(k), \alpha(k) \) (here they are considered as time-dependent and normalized with respect to the error) are derived using the following Lyapunov function [5]:

\[ L(k) = \| J(k) \|^2 + \| B(k) \|^2 + \| C(k) \|^2. \]

Then the identification error is bounded, i.e.:

\[ \Delta L(k) \leq \eta(k) |E(k)|^2 - \alpha(k) |E(k - 1)|^2 + d; \quad \Delta L(k) = L(k) - L(k - 1), \]

where all the unmodelled dynamics, the approximation errors and the perturbations are represented by the \( d \)-term, and the complete proof of that theorem and two lemmas are given in [5].

2.3. Direct and indirect adaptive neural control laws

The block-diagram of the direct adaptive neural control system is given in Fig. 1.

The control scheme contains three RTNNs. The RTNN-1 is a plant identifier, learned by the identification error \( E_i = Y_d - Y \), which estimates the state vector and the plant parameters. The RTNN-2 and RTNN-3 are feedback and feedforward neural controllers, both learned by the control error \( E_c = R - Y_d \). The control vector is a sum of both control RTNN functions \( F_{fb}, F_{ff} \), learned by the same learning procedure \( \Pi (L, M, N, Y_d, U, X, J, B, C, E) \) [6], and it is given by

\[ U(k) = -U_{fb}(k) + U_{ff}(k) = -F_{fb}[X(k)] + F_{ff}[R(k)]. \]
The block-diagram of the indirect adaptive neural control system is given in Fig. 2 [6,8].

The control scheme contains an RTNN identifier, learned by the same learning procedure $II(L, M, N, Y_d, U, X, J, B, C, E)$, which issue a parameter and state information to the sliding mode controller, designed using the methodology given in [8]. The SM control law is given by the equations:

$$U_{eq}(k) = (CB)^{-1} \left[ -CJX(k) + R(k + 1) + \sum_{i=1}^{p} \gamma_i E(k - i + 1) \right] ; \quad |\gamma_i| < 1,$$

(15)

$$U^*(k) = \begin{cases} U_{eq}(k) & \text{if } \|U_{eq}(k)\| < U_0, \\ -U_0 U_{eq}(k) / \|U_{eq}(k)\| & \text{if } \|U_{eq}(k)\| \geq U_0, \end{cases}$$

(16)

where $U_0$ is an upper bound of the control level, $p < N$ is the derivative order of the desired error model (sliding surface) and $\gamma_i$'s are parameters of the desired error model, where the right-hand side stability condition of (15) needs to be fulfilled.

3. HFNMM identifier and HFNMM controllers description

3.1. HFNMM description

Let us assume that the unknown system $y = f(x)$ generates the data $y(k)$ and $x(k)$ measured at $k, k-1, \ldots, p$, then the aim is to use this data to construct a deterministic function $y = F(x)$ that can serve as a reasonable approximation of $y = f(x)$ in which the function $f(x)$ is unknown. The variables $x = [x_1, \ldots, x_p]' \in \mathbb{R} \subset \mathbb{R}^p$ and $y \in Y \subset \mathbb{R}$ are called regressor and regressand, respectively. The variable $x$ is called an antecedent variable and the variable $y$ is called a consequent variable. In FS modeling, the function $F(x)$ is represented as a collection of IF–THEN fuzzy rules, represented by the statement:

**IF** antecedent proposition **THEN** consequent proposition.
The linguistic fuzzy model of Zadeh and Mamdani, cited in \cite{3,19} consists of rules $R_i$, where both the antecedent and the consequent are fuzzy propositions:

$$R_i: \text{If } x(k) \text{ is } A_i \text{ then } y(k) \text{ is } B_i, \quad i = 1, 2, \ldots, P$$

with $A_i$ and $B_i$ being linguistic terms (labels) defined by fuzzy sets $\mu_{A_i}(x) : X \rightarrow [0, 1]$ and $\mu_{B_i}(y) : Y \rightarrow [0, 1]$, respectively; $\mu_{A_i}(x)$, $\mu_{B_i}(y)$ are membership functions of the correspondent variables; $R_i$ denotes the $i$th rule and $P$ is the number of rules in the rule base. The model of T–S \cite{30} is a mixture between a linguistic and mathematical regression models. The rules antecedents describe the fuzzy regions in the input space. The rule consequent is crisp mathematical function of the inputs. The T–S model has the most general form

$$R_i: \text{If } x(k) \text{ is } A_i \text{ then } y_i(k) = f_i [x(k)], \quad i = 1, 2, \ldots, P. \quad (17)$$

The consequent part of the T–S model (17) could be also a dynamic state-space model. So, the T–S model could be rewritten in the form

$$R_i: \text{If } x(k) \text{ is } A_i \text{ and } u(k) \text{ is } B_i \text{ then } \begin{cases} x_i(k + 1) = J_i x_i(k) + B_i u(k), \\ y_i(k) = C_i x(k), \end{cases} \quad (18)$$

where in the antecedent part $A_i$ and $B_i$ are the above mentioned linguistic terms, in the consequent part, $x_i(k)$ is the variable associated with the $i$th sub-model state, $y_i(k)$ is the $i$th sub-model output and $J_i$, $B_i$, $C_i$ are parameters of this sub-model ($J_i$ is a diagonal matrix). The papers of Baruch et al. \cite{7,10,4} makes a step ahead proposing that the consequent function is an RTNN model, given by Eqs. (1)–(3). So the fuzzy-neural systems rule obtains the form

$$R_i: \text{If } x(k) \text{ is } J_i \text{ and } u(k) \text{ is } B_i \text{ then } y_i(k + 1) = N_i [x_i(k), u(k)], \quad i = 1, 2, \ldots, P, \quad (19)$$

where the function $y_i(k + 1) = N_i [x_i(k), u(k)]$ represents the RTNN, given by Eqs. (1)–(3), $i$ is the number of the function and $P$ is the total number of RTNN approximation functions. The biases, obtained in the process of BP learning of the RTNN model could be used to form the membership functions, as they are natural centres of gravity for each variable \cite{3}. The number of rules could be optimized using the MSE of RTNN’s learning, which is a natural learning of the RTNN model could be used to form the membership functions, as they are natural centres of gravity for each variable \cite{31}. As the local RTNN model could be learned by the local error of approximation $E_i = Y_{di} - Y_i$, rule (19) could be extended changing the neural function by learning procedure, denoted by $Y = \Pi(L, M, N, Y_d, U, X, J, B, C, E)$, and given by Eqs. (1)–(3), (6)–(9). In this case rule (19) could be rewritten in the form

$$R_i: \text{If } x(k) \text{ is } J_i \text{ and } u(k) \text{ is } B_i \text{ then } Y_i = \Pi_i(L, M, N_i, Y_{di}, U, X_i, J_i, B_i, C_i, E_i), \quad i = 1, 2, \ldots, P. \quad (20)$$

The output of the fuzzy neural multi-model system, represented by the upper hierarchical level of defuzzification is given by the following equation:

$$Y(k) = \sum_i w_i y_i(k), \quad (21)$$

where $w_i$ are weights, obtained from the membership functions \cite{31}. As it could be seen from Eq. (21), the output of the fuzzy-neural multi-model, approximating the nonlinear plant, is a weighted sum of the outputs of RTNN models, appearing in the consequent part of (20). In the case when the intervals of the variables, given in the antecedent parts of the rules, are not overlapping, the weights obtain values one and the weighted sum (21) is converted in a simple sum. This particular case will be also considered here. The weights $w_i$ depend on the form of the membership functions which is difficult to choose. We propose to augment the level of adaptation of the fuzzy-neural multi-model creating an upper hierarchical level of defuzzification which is an RTNN with inputs $y_i(k), \quad i = 1, \ldots, P$. So, Eq. (21) is extended like this:

$$Y = \Pi(L, M, N, Y_d, Y_0, X, J, B, C, E), \quad (22)$$

where the input vector $Y_0$ is formed from the vectors $y_i(k), \quad i = 1, \ldots, P$, $E = Y_d - Y$ is the error of learning and $\Pi(\cdot)$ is an RTNN learning procedure, given by Eqs. (1)–(3), (6)–(9). So, the output of the upper hierarchical defuzzification procedure (22) is a filtered weighted sum of the outputs of the T–S rules. As the RTNN is a universal
function approximator, the number of rules $P$ could be rather small, e.g., $P = 3$ (negative, zero and positive) in the case of overlapping membership functions and $P = 2$ (negative and positive), in the case of not overlapping membership functions. The stability of this HFNMM could be proven via linearization of the activation functions of the RTNN models and application of the methodology, given in [13]. As follows, in both HFNMM identification and control systems proposed, the three fuzzyfication intervals for the reference signal and the plant output are to be the same.

### 3.2. Systems identification by means of HFNMM

The systems identification is an essential part of the control systems theory and a powerful tool in the case when the plants mathematical model and their parameters are not known. In this point, the systems identification permits us to obtain plants structure, states and parameters when the output of the model follows the output of the plant. A block-diagram of the dynamic systems identification, using an HFNMM identifier is given in Fig. 3.

The structure of the entire identification system contains a Fuzzyfier, a fuzzy rule-based inference system (FRBIS), containing up to three T–S rules (20) and a defuzzyfier. The system uses an RTNN model as an adaptive, upper hierarchical level defuzzyfier (22). The local and global errors used to learn the respective RTNNs models are

$$E_i(k) = Y_{di}(k) - Y_i(k); \quad E(k) = Y_d(k) - Y_i(k).$$

(23)

The HFNMM identifier has two levels—lower hierarchical level of identification (LLI), and upper hierarchical level of identification (ULI). It is composed of three parts: (1) Fuzzyfication, where the normalized plant output signal $Y_d(k)$ is divided in three intervals (membership functions—$\mu_i$): positive $[1, -0.5]$, negative $[-1, 0.5]$ and zero $[-0.5, 0.5]$. (2) Lower level inference engine, which contains three T–S fuzzy rules, given by (20), and operating in the three intervals. The consequent part of each rule (the consequent learning procedure) has the $L, M, N_i$ RTNN model dimensions, $Y_{di}, U, E_i$ inputs and $Y_i$ (used as entry of the defuzzyfication level), $X_i, J_i, B_i, C_i$ outputs, used for control. The T–S fuzzy rule has the form

$$R_i: \text{If } Y_{di}(k) \text{ is } A_i \text{ then } Y_i = \Pi_i(L, M, N_i, Y_{di}, U, X_i, J_i, B_i, C_i, E_i), \quad i = 1, 2, 3.$$ (24)

(3) Upper level defuzzyfication, which consists of one RTNN learning procedure, doing a filtered weighted summation of the outputs $Y_i$ of the lower level RTNNs. The defuzzyfication learning procedure (22) has $L, M, N$ RTNN model dimensions, $Y_i (P = 3)$, $E$, inputs and $Y(k)$—output. The learning and functioning of both levels is independent.
The main objective of the HFNMM identifier is to issue states and parameters for the HFNNMM controller when its output follows the output of the plant with a minimum error of approximation.

3.3. Direct adaptive control by means of HFNMM

The tracking control problem consists in the design of a controller that asymptotically reduces the error between the plant output and the reference signal. The block diagram of this direct adaptive control is schematically depicted in Fig. 4.

The block diagram stress on the RTNN nature of the identification and control system. The identification part on the right contains three RTNNs, corresponding to the three rules, fired by the fuzzyfied plant output and taking part of the FRBIS HFNMM identifier, and the RTNN DF1 represents the defuzzyfier of the HFNMM identifier. The detailed structure of the HFNMM identifier could be seen in Fig. 3. The control part on the left contains three double RTNN blocks. The RTNN-Uff block represented the feedforward part of the control, corresponding to the rule fired by the fuzzyfied reference, and the RTNN-Ufb block represented the feedback part rule, fired by the fuzzyfied plant output, and its entries are the corresponding states, issued by the HFNMM identifier. The RTNN DF2 represents the defuzzyfier of the HFNMM controller. The detailed structure of the direct adaptive HFNMM controller is given in Fig. 5.

The structure of the entire control system has a Fuzzyfier, an FRBIS, containing up to six T–S rules (three for the feedforward part and three for the feedback part) and a defuzzyfier. The system uses an RTNN model as an adaptive, upper hierarchical level defuzzyfier, given by Eq. (22). The local and global errors used to learn the respective RTNNs models are

$$E_{ci}(k) = R_i(k) - Y_{di}(k); \quad E(k) = R(k) - Y_d(k).$$

The HFNMM controller has two levels—lower hierarchical level of control (LLC), and upper hierarchical level of control (ULC). It is composed of three parts: (1) Fuzzyfication, where the normalized reference signal $R(k)$ is divided in three intervals (membership functions—$\mu_i$): positive [1, $-0.5$], negative [$-1, 0.5$] and zero [$-0.5, 0.5$]. (2) Lower level inference engine, which contains six T–S fuzzy rules (three for the feedforward part and three for the feedback part), operating in the corresponding intervals. The consequent part of each feedforward control rule (the consequent learning procedure) has the $M, L, N_i$ RTNN model dimensions, $R_i, Y_{di}, E_{ci}$ inputs and $U_{fii}$, outputs used to form the total control. The T–S fuzzy rule has the form:

$$R_i: \text{If } R(k) \text{ is } B_i \text{ then } U_{fii} = \Pi_i(M, L, N_i, R_i, Y_{di}, X_i, J_i, B_i, C_i, E_{ci}), \quad i = 1, 2, 3.$$
The consequent part of each feedback control rule (the consequent learning procedure) has the \( M, L, N_i \) RTNN model dimensions, \( Y_{di}, X_i, E_{ci} \) inputs and \( U_{fbi} \), outputs used to form the total control. The T–S fuzzy rule has the form

\[
R_i: \text{If } Y_{di} \text{ is } A_i \text{ then } U_{fbi}(k) = \frac{1}{a_{fi}(k)}\left( M, L, N_i, Y_{di}, X_i, X_{ci}, I_i, B_i, C_i, E_{ci} \right), \quad i = 1, 2, 3. \tag{27}
\]

The total control corresponding to each membership function is a sum of its corresponding feedforward and feedback parts:

\[
U_i(k) = -U_{fbi}(k) + U_{fbi}(k). \tag{28}
\]

(3) Upper level defuzzyfication which consists of one RTNN learning procedure, doing a filtered weighted summation of the control signals \( U_i \) of the lower level RTNNs. The defuzzyfication learning procedure (22) has \( M, L, N \) RTNN model dimensions, \( U_i (P = 3), E_c \), inputs, and \( U(k) \)—output. The learning and functioning of both levels is independent. The main objective of the HFNMM controller is to reduce the error of control, so the plant output to track the reference signal.

### 3.4. Indirect adaptive control by means of HFNMM

The block diagram of this control is schematically depicted in Fig. 6. The block diagram stress on the RTNN nature of the identification and control system. The identification part on the right contains three RTNNs, corresponding to the three rules, fired by the fuzzyfied plant output and taking part of the FRBIS HFNMM identifier, and the RTNN DF1 represents the defuzzyfier of the HFNMM identifier. The detailed structure of the HFNMM identifier could be seen in Fig. 3. The control part on the left contains three SM control blocks. The SM control block represented a control T–S rule, fired by the fuzzyfied reference, and its entries are the corresponding states, and parameters issued by the HFNMM identifier. The RTNN DF2 represents the defuzzyfier. The detailed structure of the indirect adaptive HFNMM controller is given in Fig. 7.

The structure of the entire control system has a Fuzzyfier, an FRBIS, containing up to three T–S rules, and a defuzzyfier. The system uses an RTNN model as an adaptive, upper hierarchical level defuzzyfier, given by Eq. (22). The HFNMM controller has two levels—LLC, and ULC. It is composed of three parts: (1) Fuzzyfication, where the normalized reference signal \( R(k) \) is divided in three intervals (membership functions—\( \mu_i \)): positive \([1, -0.5]\), negative \([-1, 0.5]\), and zero \([-0.5, 0.5]\). (2) Lower Level Inference Engine, which contains three T–S fuzzy rules, operating in
the corresponding intervals. The consequent part of each SM control rule realizes the SM control (15), (16), using the state, and parameter information, issued by the corresponding RTNN of the identifier. The corresponding computational control procedure has the $M, L, N_i$ dimensions, $R_i, Y_{di}, E_{ci}, X_i, J_i, B_i, C_i$ inputs and parameters needed. The T–S fuzzy rule has the form

$$R_i: \text{If } R(k) \text{ is } B_i \text{ then } U_i = \Pi_i (M, L, N_i, R_i, Y_{di}, X_i, J_i, B_i, C_i, E_{ci}), \quad i = 1, 2, 3. \quad (29)$$

The defuzzyfication learning procedure (22) has $M, L, N$ RTNN model dimensions, $U_i (P = 3), E_c$, inputs, and $U(k)$—output. The main objective of the indirect adaptive HFNMM controller is to reduce the error of control, so the plant output to track the reference signal.
4. Simulation and experimental results

Simulation results, applying both proposed tracking control methods for nonlinear plant model, and for a 1-DOF mechanical system with friction, are given below. First the plant, used for both examples, is identified by two RTNNs operating in the positive and the negative range without overlapping and two correspondent control laws are designed. Then some experimental result of DC-motor control with the same direct HFNMM, are given. Finally some comparative simulation results of identification and control with one, two and three RTNNs and fuzzy control are shown.

4.1. Simulation results, obtained with HFNMM identification and control with \( P = 2 \) (two membership functions without overlapping)

**Example 1 (A nonlinear plant model).** Let us to consider a nonlinear single-input single-output plant, given by the following equation, taken from the paper of Ahmed [2]:

\[
y(k) = 0.9722 \ y(k - 1) + 0.3578 \ u(k - 1) - 0.1295 \ u(k - 2) - 0.3103 \ y(k - 1)u(k - 1) \\
-0.04228 \ y^2(k - 2) + 0.1663 \ u(k - 2) - 0.03259 \ y^2(k - 1) \ y(k - 2) \\
-0.3513 \ y^2(k - 1) \ u(k - 2) + 0.3084 \ y(k - 1) \ y(k - 2) \ u(k - 2) \\
+0.1087 \ y(k - 2) \ u(k - 1) \ u(k - 2).
\] (30)

The graphical simulation results of the proposed indirect adaptive two RTNN model control, are given in Fig. 8a–d.

The RTNNs learning parameters used are \( \eta = 0.2, \ \alpha = 0.01 \). The RTNNs topologies are (1, 10, 1), the period of discretization is \( T_0 = 0.001 \) s, the reference signal is \( r(k) = \sin(\pi k) + 0.5 \sin(\pi k/4) \), the control parameter \( \gamma = 0.8 \), and the desired error function order \( p = 1 \).

The result, given in Fig. 8a shows that in the beginning, there are some discrepancy between the reference signal and the output plant signal due to improper identification, which reduces after few seconds, when the learning process has converged. The final MSE\% of control is about 2.5%.

**Example 2 (A 1-DOF mechanical plant with friction control).** Let us consider a DC-motor-driven nonlinear mechanical system with friction [18], to have the following friction parameters: \( \alpha = 0.01 \) m/s, \( F_s^+ = 4.2 \) N, \( F_s^- = -4.0 \) N; \( \Delta F^+ = 1.8 \) N, \( \Delta F^- = -1.7 \) N, \( \nu_{CF} = 0.1 \) m/s; \( \beta = 0.5 \) Ns/m. Let us also consider that position and velocity measurements are taken with period of discretization \( T_0 = 0.1 \) s, the system gain \( k_0 = 8 \), the mass \( m = 1 \) kg, and the load disturbance depends on the position and he velocity, \( (d(t) = d_1q(t) + d_2v(t), d_1 = 0.25, d_2 = -0.7) \). So the discrete-time model of the 1-DOF mass mechanical system with friction is obtained in the form [8]

\[
x_1(k + 1) = x_2(k),
\] (31)

\[
x_2(k + 1) = -0.025x_1(k) - 0.3x_2(k) + 0.8u(k) - 0.1f_{r}(k),
\] (32)

\[
v(k) = x_2(k) - x_1(k),
\] (33)

\[
y(k) = 0.1x_1(k),
\] (34)

where \( x_1(k), x_2(k) \) are system states, \( v(k) \) is shaft velocity, \( y(k) \) is shaft position and \( f_{r}(k) \) is a friction force, taken from [18], with given up values of friction parameters. The graphics of the simulation results, obtained with the direct adaptive control HFNMM system, are shown in Fig. 9a–d.

The graphics in Fig. 9a compare the output of the plant with the reference which is \( r(k) = 0.8 \sin(2\pi k) \). To form the local neural controls, this reference signal is divided in two parts—positive and negative. The time of learning is 60 s. The two identification RTNNs have architectures (1, 5, 1) and the two feedback control RTNNs have architectures (5, 5, 1). The two feedforward RTNNs architectures are (1, 5, 1). The learning parameters are \( \eta = 0.1, \ \alpha = 0.2 \). Fig. 9b shows the results of identification, where the output of the plant is divided in two parts, which are identified by two RTNNs. The combined control signal and the MSE\% of control are given in Fig. 9c, d. As it could be seen from the last graphics, the MSE\% rapidly decreases, and reached values below 1%. The graphics of the simulation results, obtained with the indirect adaptive HFNMM control system, are shown in Fig. 10a–d.

Fig. 10a compares the output of the plant with the reference signal \( r(k) = \sin(\pi k) + 0.5 \sin(\pi k/4) \), which is divided in two parts—positive and negative. The time of learning is 100 s. The two identification and the two FF control RTNNs
Fig. 8. Graphical results of simulation using indirect adaptive fuzzy-neural multi-model control (example 1): (a) comparison of the output of the plant with the reference signal; (b) comparison of the output of the plant with the outputs of the identification RTNNs; (c) combined control signal and (d) mean squared error of control (MSE).

have topologies (1, 5, 1). The learning rate parameters are $\eta = 0.001$, $\alpha = 0.01$, and the control parameter is $\gamma = 0.1$. Fig. 10b shows the results of identification, where the output of the plant is divided in two parts, identified by two RTNNs. The state and parameter information issued by the identification multi-model is used to design a linear SM control law. The combined control signal and the MSE% of control are given in Fig. 10c, d. As it could be seen from the last graphics, the MSE% rapidly decreases, and reached values below 2%.

4.2. Experimental results, obtained with HFNMM identification and control with $P = 2$ (two membership functions without overlapping)

In this section, the effectiveness of the multi-model scheme is illustrated by a real-time DC-motor identification and position control [5], using two RTNNs (positive and negative), for the systems output, control, and reference signals. A 24 V, 8 A DC motor, driven by a power amplifier and connected by a data acquisition control board (Multi-Q™) with the PC, has been used. The RTNN was programmed in MatLab™-Simulink™ and WinCon™, which is a real-time Windows 95 application that runs Simulink generated code using the real-time Workshop to achieve digital real-time control on a PC. The load is charged and discharged on the DC-motor shaft by means of electrically switched clutch.

The first experiment corresponds to an open-loop shaft velocity real-time identification of the DC-motor dynamics. The fuzzy-neural multi-model contains two RTNN with topologies (1, 5, 1). The learning rate parameters are $\eta = 0.8$ and $\alpha = 1.0$. The sampling period is $T_0 = 0.01$. The DC-motor input is a sum of two sinusoid functions $U(k) = 0.1 \sin(6\pi k) + 0.1 \sin(1.6\pi k + 15)$. The results of systems identification are shown in Fig. 11a–f.
Fig. 9. Graphical results of simulation using direct adaptive fuzzy-neural multi-model control (example 2): (a) comparison of the output of the plant and the reference signal; (b) comparison of the output of the plant and the outputs of the identification RTNNs; (c) combined control signal and (d) mean squared error of control (MSE).

Fig. 11a shows the sum of RTNN-1 and RTNN-2 outputs, used for the positive and the negative output model identification. It is seen that the multi-model output signal follows the real speed response of the DC motor. Fig. 11b shows both RTNN-1 and RTNN-2 model outputs that form the multi-model identification system. It could be seen in the figure, that each RTNN follow its corresponding DC-motor speed signal (positive or negative). Fig. 11e shows the instantaneous identification error of the RTNN-1 and RTNN-2, respectively. The second experiment is the closed-loop DC-motor shaft position identification, where the input is generated by a PD control of system error and the reference signal of the closed loop system is a sum of four sinusoid functions \( r(k) = 20 \sin(6\pi k) + 20 \sin(1.6\pi k + 15) + 20 \sin(1.2\pi + 30) + 20 \sin(0.8\pi + 45) \). The RTNN parameters and RTNN topologies are the same as in the first case. Fig. 11c compares the real-time closed-loop position response of the DC motor and the output of the multi-model identification system. Fig. 11d shows the individual responses of both RTNNs (1 and 2), and Fig. 11f shows the instantaneous identification error of the RTNN-1 and RTNN-2, respectively. The third experiment is the closed-loop direct adaptive control of the DC-motor shaft position. The control signal is designed in two parts—feedback and feedforward. For sake of simplicity, the feedback part is realized as a P-controller \( (K_p = 5) \) and only a feedforward part of the fuzzy-neural control scheme is applied which allow us to omit the systems identification in real time. So, the neural multi-model feedforward controller is realized by means of two RTNNs—positive and negative. The graphics of the experimental results obtained are given in Fig. 12a–e.

Fig. 12a, b compare the DC-motor shaft position with the reference signal \( (r(k) = 1.5 \sin(0.2\pi k)) \) in the absence of load (Fig. 12a, 0–45 s) and in the presence of load (Fig. 12b, 70–150 s). The control signal and the MSE% of reference tracking are given in Fig. 12c, d, respectively, for the complete time of the experiment (0–200 s). The architectures of
the two feedforward RTNNs are (1, 5, 1). The learning rate parameters are \( \eta = 0.003, \alpha = 0.0001 \) and the period of discretization is \( T_0 = 0.001 \) s. The MSE\% exhibits fast convergence and rapidly reached values below 1%. For sake of comparison, the same experiment was repeated for DC-motor position control with PD controller (\( K_p = 3, K_d = 0.05 \)). The results obtained without load are almost equal to that of Fig. 12a, but when the load is charged on DC-motor shaft, a lack of tracking precision is observed (see Fig. 12e, 45–100 s). The obtained values of the MSE\% for both control experiments (with a fuzzy-neural multi-model control \(-0.286\%\), and with a PD control \(-0.761\%) show that the precision of tracking of the fuzzy-neural multi-model control is about two times greater than that obtained by means of the PD control. Furthermore, the fuzzy-neural multi-model control could adapt to a load variation and the PD control needs gains update.

4.3. Comparative simulation results, obtained with direct HFNMM identification and control with \( P = 3 \) (three membership functions with overlapping), with one RTNN identification and control, and with the fuzzy control

Finally, a respective comparative results of direct adaptive plant control, obtained using only one RTNN, and that using HFNMM control are given in Fig. 13a–d, using the same 1-DOF mechanical plant model.

The topology of the identification and FF control RTNNs is (1, 5, 1) and that of the FB control RTNN is (5, 5, 1). The topology of the defuzzyfication RTNN is (1, 3, 1). The results of these neural controls are compared with the fuzzy control (Fig. 13e, f). Fig. 13b–f shows the MSE\% of control which final values are 0.41\% for the direct adaptive HFNMM control, 2.7\% for the control with single RTNNs, and 5.8\% for the fuzzy control. The learning rate parameters are \( \eta = 0.01, \alpha = 0.9 \) and the period of discretization is \( T_0 = 0.01 \) s. The reference signal is \( r(k) = \text{sat}[0.5 \sin(\pi k) + 0.5 \sin(\pi k/2)] \). The level of saturation is \( \pm 0.8 \).
Fig. 11. Experimental results of a DC-motor velocity and position real-time identification, using two RTNNs fuzzy-neural multi-model. (a) Comparison of the DC-motor velocity (continuous line) and the sum of the outputs of both RTNNs (dashed line). (b) Comparison of the DC-motor velocity (continuous line) and the outputs of both RTNNs—positive and negative (dashed line). (c) Comparison of the DC-motor position (continuous line) and the sum of the outputs of both RTNNs (dashed line). (d) Comparison of the DC motor position (continuous line) and the outputs of both RTNNs—positive and negative (dashed line). (e) Instantaneous errors of velocity identification by the positive and negative RTNNs. (f) Instantaneous errors of position identification by the positive and negative RTNNs.
Fig. 12. Experimental results of a DC-motor position control using direct adaptive neural multi-model control scheme. Comparison of the output of the plant and the reference signal: (a) without load (0–45 s); (b) with load (70–150 s); (c) combined control signal (0–200 s); (d) mean squared error of control and (e) comparison of the output of the plant and the reference signal for control with PD controller and load on a DC-motor shaft (45–100 s).
Fig. 13. Comparative trajectory tracking control results obtained with: a single RTNN feedforward/feedback direct adaptive control, with an HFNMM feedforward/feedback direct adaptive control; and with a fuzzy control. (a) Comparison of the reference signal and the output of the plant using single RTNN controllers; (b) MSE of a single RTNN controller; (c) comparison of the reference signal and the output of the plant using HFNMM control, (d) MSE of an HFNMM control, (e) comparison of the reference signal and the output of the plant using fuzzy control and (f) MSE of a fuzzy control.

From the graphics of Fig. 13 a–d, we could see that the direct adaptive HFNMM feedforward—feedback control is better than that using single RTNNs. From the graphics of Fig. 13e, f, we could see that the fuzzy control is worse with respect to the neural control, especially when the friction parameters changed.

5. Conclusions

An HFNMM, containing fuzzy rules with procedural consequent parts, is proposed to be used for identification and control of complex nonlinear plants. Two control schemes (direct and indirect) using an HFNMM has been experimented and compared with a respective single-RTNN control and a fuzzy control. The proposed identification and control methods have the following advantages with respect to the conventional methods of intelligent fuzzy and neural control: (1) the number of T–S rules used is lower, (2) the level of adaptation is higher, (3) the resistance of noise and the robustness is superior due to the error correction (BP) learning of RTNNs applied. The comparison of identification results for a 1-DOF mechanical system with friction shows that the FNHMM identifier has a better performance (0.08% vs. 0.27% for MSE% of identification) with respect to the identification using only one RTNN. For the first control scheme the MSE% of control is 1.5% vs. 2.3%, and for the second control scheme—it is 0.41% vs. 2.7%, with respect to the control using single RTNNs. The fuzzy control has the worse precision of MSE% = 5.8%.
References