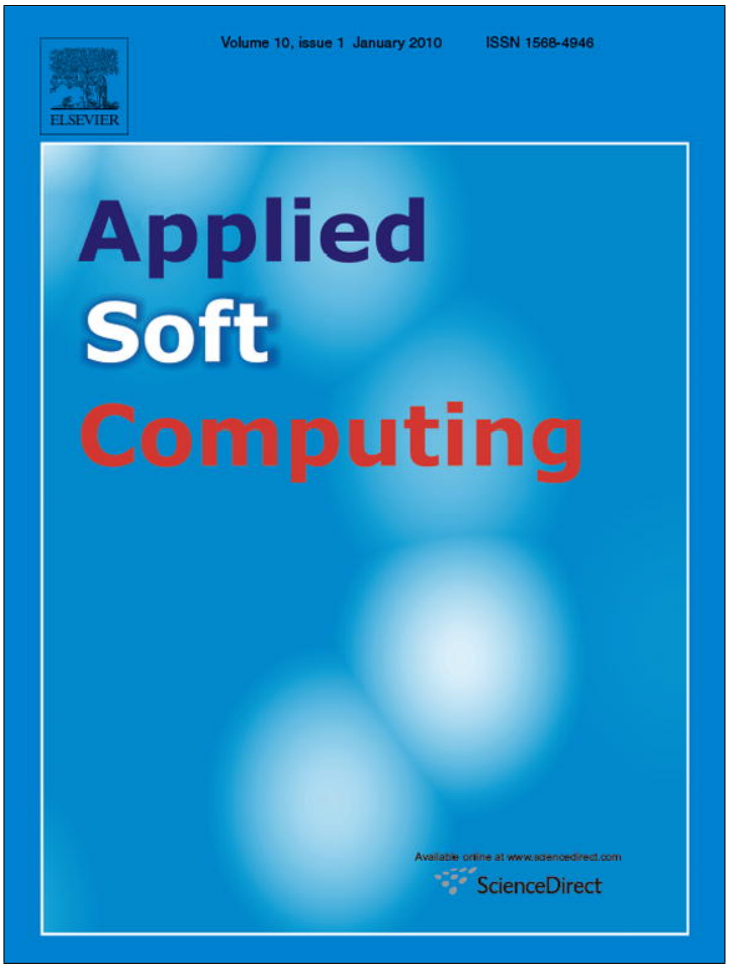
Volume 10, issue 1 January 2010
ISSN 1568-4946
ELSEVIER
Applied
Soft
Computing
Available online at www.sciencedirect.com
ScienceDirect

Applied Soft Computing 10 (2010) 267–275

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Unsupervised adaptive neural-fuzzy inference system for solving differential equations

Hadi Sadoghi Yazdi *, Reza Pourreza

Computer Engineering Department, Ferdowsi University of Mashhad, Mashhad, Iran

ARTICLE INFO

Article history:
Received 31 May 2007
Received in revised form 23 July 2009
Accepted 26 July 2009
Available online 4 August 2009

Keywords:
Unsupervised neuro-fuzzy
Differential equation
Neural network
Fuzzy inference system

ABSTRACT

There has been a growing interest in combining both neural network and fuzzy system, and as a result, neuro-fuzzy computing techniques have been evolved. ANFIS (adaptive network-based fuzzy inference system) model combined the neural network adaptive capabilities and the fuzzy logic qualitative approach. In this paper, a novel structure of unsupervised ANFIS is presented to solve differential equations. The presented solution of differential equation consists of two parts; the first part satisfies the initial/boundary condition and has no adjustable parameter whereas the second part is an ANFIS which has no effect on initial/boundary conditions and its adjustable parameters are the weights of ANFIS. The algorithm is applied to solve differential equations and the results demonstrate its accuracy and convince us to use ANFIS in solving various differential equations.

1. Introduction

Artificial neural network is a favorable technique to solve optimization problems because it can simulate the operations of the brain and uses parallel processing to save computational time [1]. Fuzzy logic approach is another intelligent computing tool which is competent for applying to wide variety of problems. Primary aim of Lotfi Zadeh who introduced the notion of a "fuzzy set" in 1965 was to set up a formal framework for the representation and management of vague and uncertain knowledge. Neural Networks are demonstrated to have powerful capability of expressing relationship between input–output variables. In fact it is always possible to develop a structure that approximates a function with a given precision. However, there is still distrust about the neural networks identification capability in some applications [2]. Fuzzy set theory plays an important role in dealing with uncertainty in plant modeling applications.

Recently, there has been a growing interest in combining both these approaches, and as a result, neuro-fuzzy computing techniques have been evolved. Neuro-fuzzy systems are fuzzy systems, which use neural networks theory in order to determine their properties (fuzzy sets and fuzzy rules) by processing data samples [3]. Neuro-fuzzy integrates to synthesize the merits of both neural networks and fuzzy systems in a complementary way to overcome their disadvantage. The fusion of neural network and fuzzy logic in neuro-fuzzy models possess both low-level learning and computational power of neural networks and advantages of high-level human like thinking of fuzzy systems. ANFIS (adaptive network-based fuzzy inference system) model combined the neural network adaptive capabilities and the fuzzy logic qualitative approach.

ANFIS presented by Jang [4]. It has attained its popularity due to a broad range of useful applications in such diverse areas in recent years as optimization of fishing predictions [5], vehicular navigation [6], identify the turbine speed dynamics [7], radio frequency power amplifier linearization [8], microwave application [9], image denoising [10,11], prediction in cleaning with high pressure water [12], sensor calibration [13], fetal electrocardiogram extraction from ECG signal captured from mother [14], identification of normal and glaucomatous eyes [15]. All these works show that ANFIS is a good universal approximator, predictor, interpolator and estimator and demonstrate that ANFIS has the approximation capabilities of neural networks and any nonlinear function of several inputs and outputs can be easily constructed with ANFIS. The summarized advantages of the ANFIS technique is listed below.

- Real-time processing of instantaneous system input and output data's. This property helps using of this technique for many operational researches problems.
- Offline adaptation instead of online system-error minimization, thus easier to manage and no iterative algorithms are involved.
- System performance is not limited by the order of the function since it is not represented in polynomial format.

* Corresponding author.
E-mail address: sadoghi@sttu.ac.ir (H.S. Yazdi).

1568-4946/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.07.006

- Fast learning time.
- System performance tuning is flexible as the number of membership functions and training epochs can be altered easily.
- The simple if–then rules declaration and the ANFIS structure are easy to understand and implement.

In the other side, numerous problems in science and engineering can be converted to a set of differential equations. Basic numerical methods can be achieved to solve differential equations such as the finite difference method, finite-element method, finite volume method and the boundary element method. Beside these basic methods, some researchers have utilized the approximation properties of neural networks to solve differential equations because solving ordinary and partial differential equations can be achieved to a good degree of precision by an artificial neural network. However there is no work on using ANFIS to solve differential equations and the dynamic behavior and approximation capability of ANFIS motivated us in this study to use it in solving differential equations. The main objectives of this paper is to show that by reformulation of differential equations, a novel unsupervised ANFIS can be used to solve differential equations numerically.

Since our method is close to neural network methods, the neural network-based approaches for solving differential equations are investigated in the following section.

1.1. Related works

The basic ideas of solving differential equations by neural networks were presented by Lagaris et al. [16]. Let the differential equations to be solved be given by (1)

$$G(x, \psi(x), \nabla\psi(x), \nabla^2\psi(x), \ldots) = 0 \quad x \in \bar{D} \subseteq R^n, \tag{1}$$

where $\psi(x)$ denotes the solution, G is the function that defining the structure of the differential equation, $\bigtriangledown$ is some differential operator, and $\bar{D}$ is the problem domain. The basic idea, called collocation method, is to discretize the domain $\bar{D}$ over a finite set of points D. Thus (1) becomes a system of equations. An approximation of the solution $\psi(x)$ is given by the trial solution $\psi_t(x)$. As a measure for the degree of fulfillment of the original differential equation (1) an error function similar to the mean squared error is defined:

$$E = \frac{1}{|D|}\sum_{x_t \in D}[G(x_i, \psi_t, \nabla\psi_t, \nabla^2\psi_t, \ldots)]^2 \tag{2}$$

Therefore, finding an approximation of the solution of (1) is equal to finding a function which minimizes the error E. Since multilayer feed forward neural networks are universal approximators the trial solution $\psi_t(x)$ can be represented by such an artificial neural network as (3) [17].

$$\psi_t(x) = A(x) + F(x, N(x, p)) \tag{3}$$

where $A(x)$ contains no adjustable parameter and satisfies the boundary conditions and $F(x,N(x,p))$ is a single output feed forward neural network with the input vector x and parameters p. In case of a given network architecture the problem is reduced to finding a configuration of weights that minimizes (2). As E is differentiable with respect to the weights for most differential equations, efficient, gradient-based learning algorithms for artificial neural networks can be employed for minimizing (2). The history of solving differential equation using neural networks is reviewed in the rest of this section.

In [18] Hüsken, and Goerick have been focused on the choice of a set of initial weights using an evolutionary solving of a differential equation with variable boundary conditions. Smaoui and Al-Enezi analyzed dynamics of two nonlinear partial differential equations known as the Kuramoto–Sivashinsky (K–S) equation and the two-dimensional Navier–Stokes (N–S) equations using Karhunen–Loeve (K–L) decomposition and artificial neural networks [19]. In [20], Brause used differential equations for modeling of biochemical pathways and these equations were solved using neural networks. In [21], Hea et al. used feed forward neural network with the extended back propagation algorithm to solve a class of first-order partial differential equations for input-to-state linearizable or approximate linearizable systems.

Also Manevitz et al. presented basic learning algorithms and the neural network model to the problem of mesh adaptation for the finite-element method to solve time-dependent partial differential equations. Time series prediction via the neural network methodology was used to predict the areas of "interest" in order to obtain an effective mesh refinement at the appropriate times [22]. Leephakpreeda presented fuzzy linguistic model in neural network to solve differential equations and applied it as universal approximators for any nonlinear continuous functions [23].

In [24], Malek and Beidokhti presented a hybrid method based on optimization techniques and neural networks methods for solving high order ordinary differential equations. They proposed a new solution method for the approximated solution of high order ordinary differential equations using innovative mathematical tools and neural-like systems of computation. Hybrid method could result in improved numerical methods for solving initial/boundary value problems without using pre-assigned discretization points.

In another work, Mai-Duy and Tran-Cong presented mesh-free procedures for solving linear differential equations, ordinary differential equations and elliptic partial differential equations based on multi quadric radial basis function networks [25]. Also Jianyu et al. in [26] described a neural network for solving partial differential equations which activation functions of the hidden nodes were the radial basis functions (RBF) whose parameters were learnt by a two-stage gradient descent strategy. Also solving differential equation using neural network was applied to real problems such as a non-steady fixed bed non-catalytic solid/gas reactor [27].

In the aforementioned works, some notes are observed as follows:

- Solving differential equation includes ordinary or partial or high order form using neural networks.
- Some researchers performed logistic function using neural networks for solving differential equations.
- Application of neural network-based differential equation in solving real world problems.

Hence we were encouraged to present a new unsupervised neuro-fuzzy inference system to solve differential equation.

The paper is organized as follows. The architecture of adaptive neuro-fuzzy inference system is explained in Section 2. Section 3 is devoted to solve of differential equation using unsupervised ANFIS algorithm. Experimental results are discussed in Section 4 and in final section conclusions are presented.

2. Adaptive neuro-fuzzy inference system (ANFIS) architecture

Neuro-fuzzy systems are fuzzy systems, which use NNs to determine their properties (fuzzy sets and fuzzy rules) by processing data samples. Neuro-fuzzy integrates to synthesize the merits of both NN and fuzzy systems in a complementary way to overcome their disadvantage. The fusion of a NN and fuzzy logic in neuro-fuzzy models possess both low-level learning and

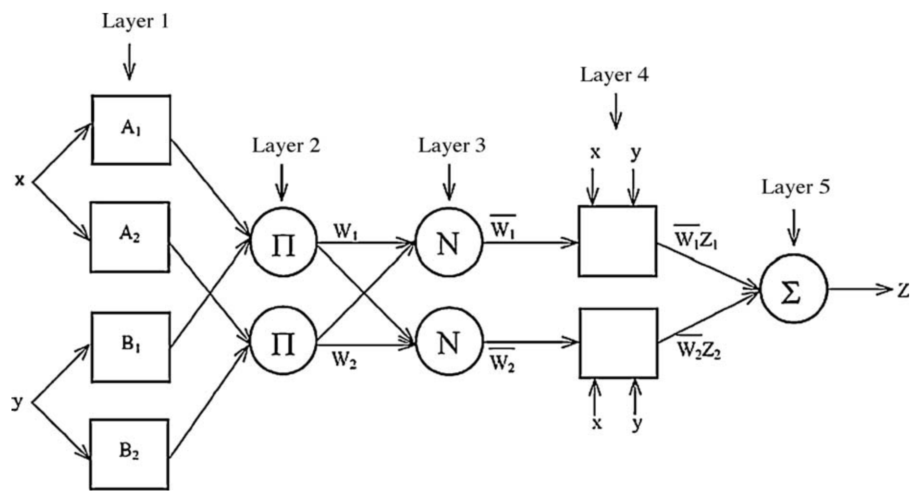

Fig. 1. ANFIS architecture (Π, N, Σ are defined in (7), (8) and (10), respectively).

computational power of NNs and advantages of high-level human like thinking of fuzzy systems. For identification, hybrid neuro-fuzzy system called ANFIS combines a NN and a fuzzy system together. ANFIS has been proved to have significant results in modeling nonlinear functions. In ANFIS, the membership functions (MFs) are extracted from a data set that describes the system behavior. The ANFIS learns features in the data set and adjusts the system parameters according to given error criterion. In a fused architecture, NN learning algorithms are used to determine the parameters of fuzzy inference system.

A typical architecture of ANFIS is shown in Fig. 1, in which a circle indicates a fixed node, and a square indicates an adaptive node. For simplicity, we consider two inputs x, y and one output z in the fuzzy inference system (FIS). The ANFIS used in this paper implements a first-order Sugeno fuzzy model. Among many fuzzy inference systems, the Sugeno fuzzy model is the most widely used for its high interpretability and computational efficiency, and built-in optimal and adaptive techniques. For example for a first-order Sugeno fuzzy model, a common rule set with two fuzzy if-then rules can be expressed as (4).

$$\begin{array}{ll} \text{Rule 1}: & \text{If } x \text{ is } A_1 \text{ and } y \text{ is } B_1, \text{ then} \\ & z_1 = p_1x + q_1y + r_1 \\ \text{Rule 2}: & \text{If } x \text{ is } A_2 \text{ and } y \text{ is } B_2, \text{ then} \\ & z_2 = p_2x + q_2y + r_2 \end{array} \tag{4}$$

where A_i, B_i (i = 1, 2) are fuzzy sets in the antecedent, and p_i, q_i, r_i (i = 1, 2) are the design parameters that are determined during the training process. As in Fig. 1, the ANFIS consists of five layers.

Layer 1, every node i in this layer is an adaptive node with a node function:

$$\begin{array}{ll} O_i^1 = \mu_{A_i}(x), & i = 1,2 \\ O_i^1 = \mu_{B_i}(y), & i = 3,4 \end{array} \tag{5}$$

where x, y are the input of node i, and $\mu_{A_i}(x)$ and $\mu_{B_i}(y)$ can adopt any fuzzy membership function (MF). In this paper, Gaussian MFs are used:

$$\text{Gaussian}(x, c, \sigma) = e^{-(1/2)((x-c)/\sigma)^2} \tag{6}$$

where c is center of Gaussian membership function and σ is standard deviation of this cluster.

Layer 2, every node in the second layer represents the ring strength of a rule by multiplying the incoming signals and forwarding the product as:

$$O_i^2 = \omega_i = \mu_{A_i}(x) \cdot \mu_{B_i}(y), \quad i = 1,2 \tag{7}$$

Layer 3, the ith node in this layer calculates the ratio of the ith rule's ring strength to the sum of all rules' ring strengths:

$$O_i^3 = \varpi_i = \frac{\omega_i}{\omega_1 + \omega_2}, \quad i = 1,2 \tag{8}$$

where ϖ_i is referred to as the normalized ring strengths.

Layer 4, the node function in this layer is represented by

$$O_i^4 = \varpi_i z_i = \varpi_i (p_i x + q_i y + r_i), \quad i = 1,2 \tag{9}$$

where ϖ_i is the output of layer 3, and $\{p_i, q_i, r_i\}$ are the parameter set. Parameters in this layer are referred to as the consequent parameters.

Layer 5, the single node in this layer computes the overall output as the summation of all incoming signals:

$$O_1^5 = \sum_{i-1}^{2} \varpi_i z_i = \frac{\omega_1 z_1 + \omega_2 z_2}{\omega_1 + \omega_2} \tag{10}$$

It is seen from the ANFIS architecture that when the values of the premise parameters are fixed, the overall output can be expressed as a linear combination of the consequent parameters:

$$\begin{aligned} z = {} & (\varpi_1 x)\, p_1 + (\varpi_1 y) q_1 + (\varpi_1) r_1 + (\varpi_2 x)\, p_2 + (\varpi_2 y) q_2 \\ & + (\varpi_2) r_2 \end{aligned} \tag{11}$$

The hybrid learning algorithm [4,28] combining the least square method and the back propagation (BP) algorithm can be used to solve this problem. This algorithm converges much faster since it reduces the dimension of the search space of the BP algorithm. During the learning process, the premise parameters in layer 1 and the consequent parameters in layer 4 are tuned until the desired response of the FIS is achieved. The hybrid learning algorithm has a two-step process. First, while holding the premise parameters fixed, the functional signals are propagated forward to layer 4, where the consequent parameters are identified by the least square method. Second, the consequent parameters are held fixed while the error signals, the derivative of the error measure with respect to each node output, are propagated from the output end to the input end, and the premise parameters are updated by the standard BP algorithm.

3. Unsupervised ANFIS for solving differential equations

A linear differential equation (DE) with constant coefficients can be expressed to following form,

$$a_n\frac{d^n y(t)}{dt^n}+a_{n-1}\frac{d^{n-1}y(t)}{dt^{n-1}}+\cdots+a_0 y(t)=\nu_o(t),\quad t\in[a,b] \tag{12}$$

where $a_n,\ldots,a_0$ are constant coefficients and $[a, b]$ is the problem domain. $n-1$ necessary initial conditions or boundary conditions for solving above DE are

$$\begin{array}{l} y(0)=y_0^0, y^{(1)}(0)=y_0^{(1)},\ldots,y^{(n-1)}(0)=y_0^{(n-1)} \\ \text{or} \\ y(t_0)=y_{t_0}, y(t_1)=y_{t_1},\ldots,y(t_n)=y_{t_n} \end{array} \tag{13}$$

As has been pointed in [24]; a trial solution for the above differential equation is like (14).

$$\begin{aligned} y_p(t) &= f_i(t,y_0^{(0)},y_0^{(1)},\ldots,y_0^{(n-1)})+g(h_i,u) \\ &= f_i(t,C)+g(h_i,u) \end{aligned} \tag{14}$$

where $f_i(t, C)$ is a function for satisfaction of initial/boundary conditions, C is same initial condition and $g(h_i, u)$ is a function which is zero in initial points and is u in other points. u is an unsupervised adaptive neuro-fuzzy inference system and plays a highly important role, principally u is the main answer without contemplate of initial points. An interesting point is that u is expressed in the form of fuzzy system and is tuned using hybrid learning algorithm including the least square method and the back propagation algorithm. h_i is used to suppress the $g(h_i, u)$ term in initial/boundary points. Hence an easy and suitable form of $g(h_i, u)$ is $h_i u$. $f_i(t, C)$ and h_i take different forms depending on the initial/boundary conditions and the order of differential equations and there is no clear procedure to choose the most appropriate ones. The selection of $f_i(t, C)$ and h_i for several types of DEs is explained in [16], however we repeat some ordinary equations here to make it easy for the readers to follow the procedure of solving DE using ANFIS.

Consider the first-order DE in (15).

$$\frac{dy(t)}{dt}=\nu_o(y,t),\quad t\in[0,1],\ \ y(0)=A \tag{15}$$

hence

$$f_i(t,C)=A, h_i=t\Rightarrow y_p(t)=A+tu \tag{16}$$

Now consider the following second-order DE.

$$\frac{dy(t)}{dt}=\nu_o\left(\frac{dy}{dt},y,t\right),\quad t\in[0,1] \tag{17}$$

The trial solution of this DE is written in two cases. In the first case these initial conditions are considered: $y(0)=A$ and $(d\{y(0)\}/dt)=A'$. So

$$f_i(t,C)=A+A't, h_i=t^2\Rightarrow y_p(t)=A+A't+t^2u \tag{18}$$

In the second case these boundary conditions are considered: $y(0)=A$ and $y(0)=B$. Therefore

$$\begin{aligned} f_i(t,C)&=A(1-t)+Bt, h_i=t(1-t)\Rightarrow y_p(t) \\ &=A(1-t)+Bt+t(1-t)u \end{aligned} \tag{19}$$

The same procedure is performed to find the trial solution of higher order ordinary differential equations.

Pursuing the procedure of solving DE, (14) is substituted in (12) and we can write

$$a_n\frac{d^n y_p(t)}{dt^n}+a_{n-1}\frac{d^{n-1}y_p(t)}{dt^{n-1}}+\cdots+a_0 y_p(t)=\nu_o(t) \tag{20}$$

Then putting $y_p(t)=f_i(t, C)+h_i u$ (20) becomes

$$\begin{aligned} &\hat{f}_i(t,C)+b_n\frac{d^n u_p(y_p,W,B)}{dt^n}+b_{n-1}\frac{d^{n-1}u_p(y_p,W,B)}{dt^{n-1}}+\cdots \\ &\quad+b_0 u_p(y_p,W,B)=\nu_o(t) \end{aligned} \tag{21}$$

where $\hat{f}_i(t,C)=a_n(d^n f_i(t,C)/dt^n)+a_{n-1}(d^{n-1}f_i(t,C)/dt^{n-1})+\cdots+a_0 f_i(t,C)$, b_i are coefficients which are generally functions of t and are related to the effect of h_i. Also W is between layers weights in ANFIS as shown in Fig. 1 and B includes parameter of $\{p_i, q_i, r_i\}$ and input membership parameters. Finally, we can obtain desired output of $u_p(y_p, W, B)$ from (21) for learning of ANFIS as follows:

$$\begin{aligned} u_p(y_p,W,B)=-\frac{1}{b_0}\Bigg(\nu_o(t)-\Bigg(\hat{f}_i(t,C)+b_n\frac{d^n u_p(y_p,W,B)}{dt^n} \\ +b_{n-1}\frac{d^{n-1}u_p(y_p,W,B)}{dt^{n-1}}+\ldots\Bigg)\Bigg) \end{aligned} \tag{22}$$

Already, we have acquired an equation that can be used to calculate the desired outputs to random inputs. These input–

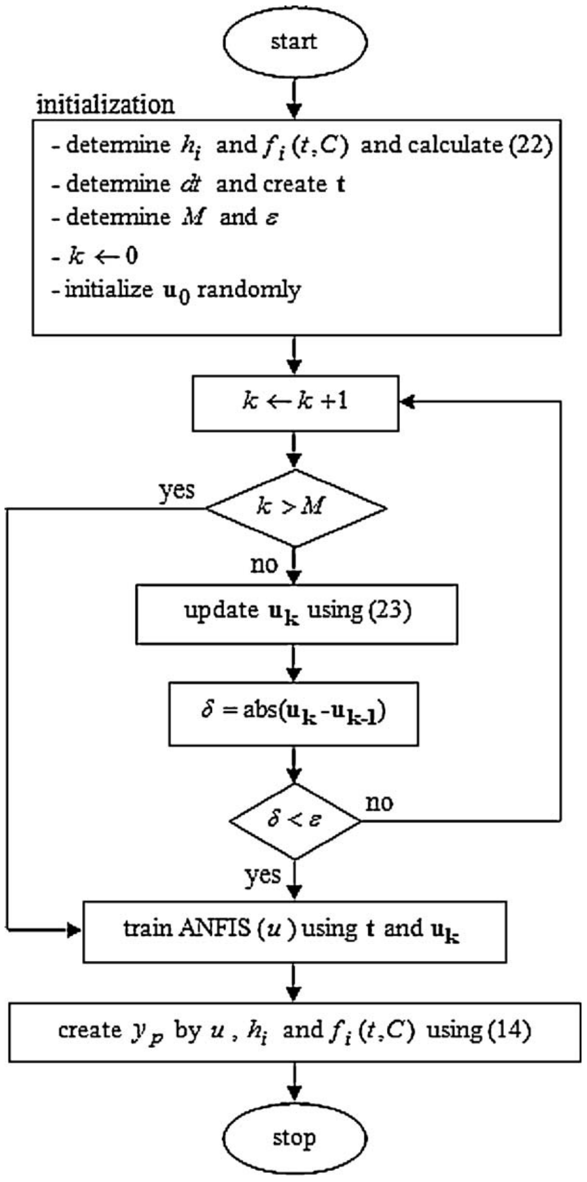
start
initialization
- determine h_i and $f_i(t,C)$ and calculate (22)
- determine dt and create t
- determine M and ε
- $k \leftarrow 0$
- initialize $\mathbf{u}_0$ randomly
$k \leftarrow k+1$
yes
$k > M$
no
update $\mathbf{u_k}$ using (23)
$\delta = \text{abs}(\mathbf{u_k}-\mathbf{u_{k-1}})$
$\delta < \varepsilon$
no
yes
train ANFIS (u) using t and $\mathbf{u_k}$
create y_p by u, h_i and $f_i(t,C)$ using (14)
stop

Fig. 2. Flowchart of the proposed algorithm.

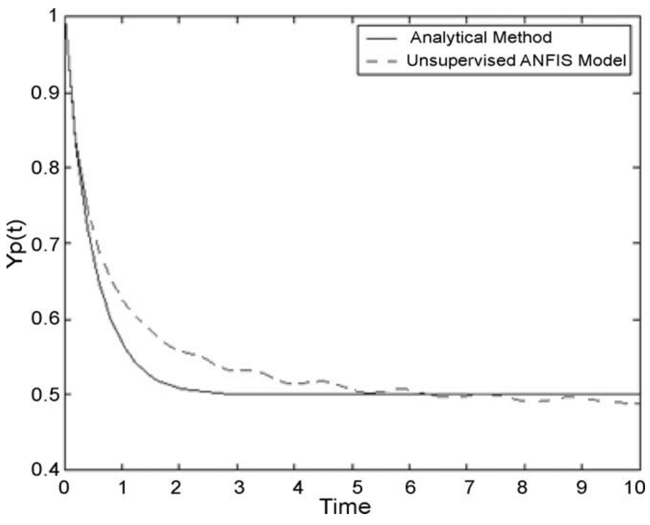

Fig. 3. Responses collation about first-order differential equation with constant excitation in Example 1.

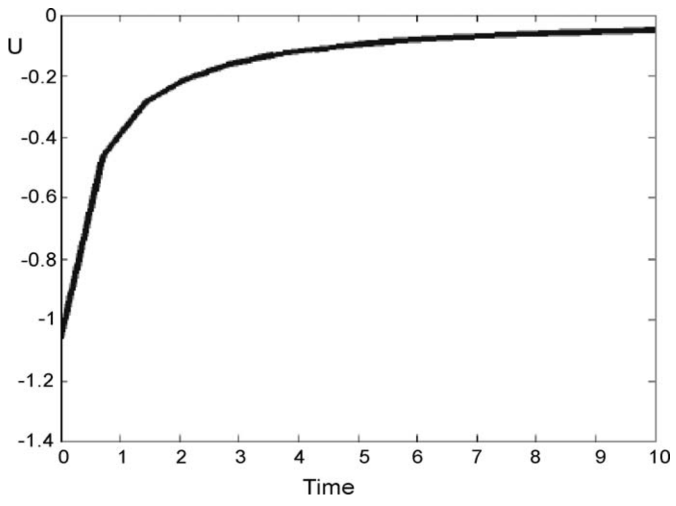

Fig. 4. ANFIS output after convergence for Example 1.

output pairs are finally used to train the unsupervised ANFIS. This process is explained in details below.

Given a differential equation, $f_i(t, C)$ and h_i should be determined as explained above as well as $u_p(y_p, W, B)$ using (22). Then we must acquire some learning samples to train ANFIS, therefore an input vector ($\mathbf{t}$) is generated so that it covers the problem domain uniformly with time step (dt). Since the implemented ANFIS is unsupervised, the output vector ($\mathbf{u}$) to the random input must be calculated automatically. We have utilized an iterative algorithm to calculate the referred outputs. To do so, $\mathbf{u}_0$ is initialized randomly and is updated using (23).

$$\mathbf{u_{k+1}} = -\frac{1}{b_0}\left(v_o(\mathbf{t}) - \left(\hat{f}_i(\mathbf{t}, C) + b_n\frac{d^n\mathbf{u_k}}{dt^n} + b_{n-1}\frac{d^{n-1}\mathbf{u_k}}{dt^{n-1}} + \ldots\right)\right). \tag{23}$$

The derivatives of $\mathbf{u}$ in (23) are calculated numerically; the first derivative is the first-order difference of $\mathbf{u}$ divided by dt, the second derivative is the second-order difference of $\mathbf{u}$ divided by dt^2 and so on.

The iteration is stopped when the stopping criteria are met. The criteria include a small difference between $\mathbf{u_{k+1}}$ and $\mathbf{u_k}$ or a large number of iteration. Hereby some input–output pairs are generated and in the next step, ANFIS is generated and trained according to the available learning samples. The final result is achieved, combining ANFIS (u), $f_i(t, C)$ and h_i according to (14).

The algorithm flowchart for solving differential equation using unsupervised ANFIS is depicted in Fig. 2.

In Fig. 2, M is the maximum number of iterations and ε is the maximum acceptable error. As have mentioned before, a hybrid

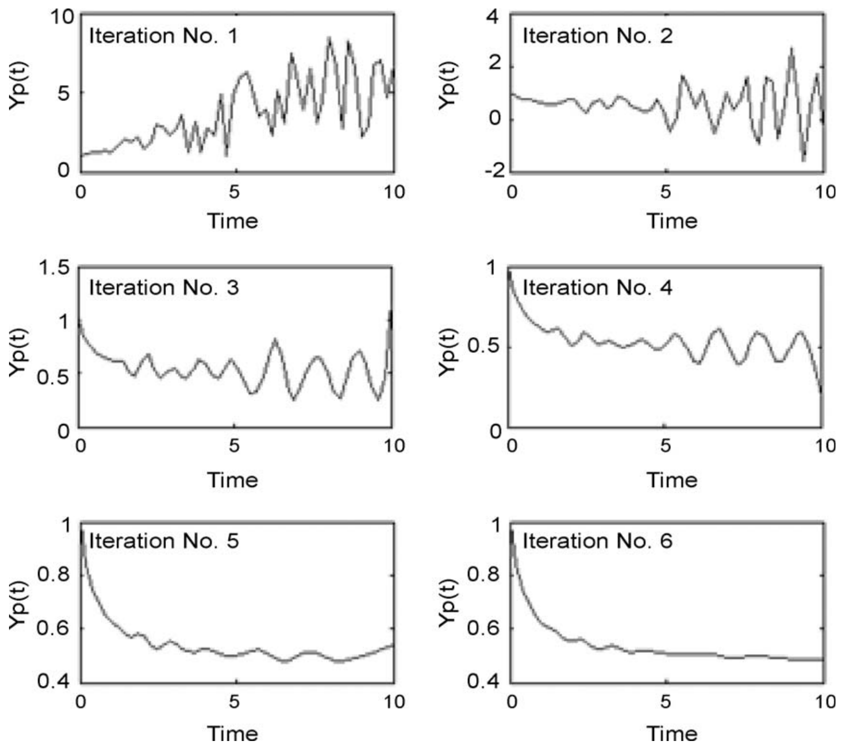

Fig. 5. Training procedure in Example 1.

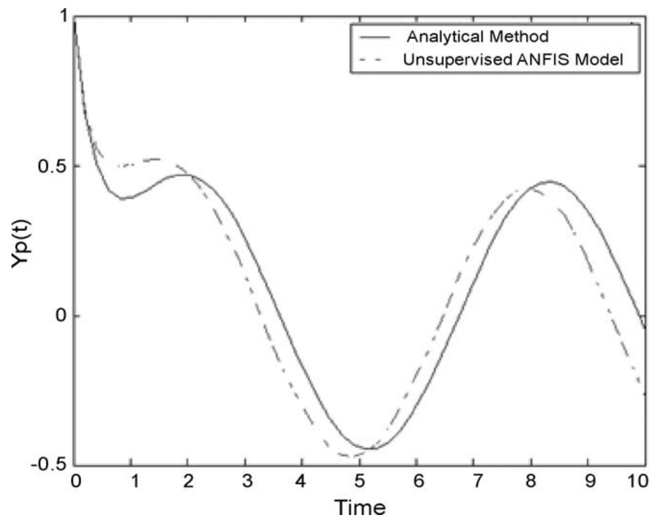

Fig. 6. Comparison of analytical and the proposed methods in Example 2.

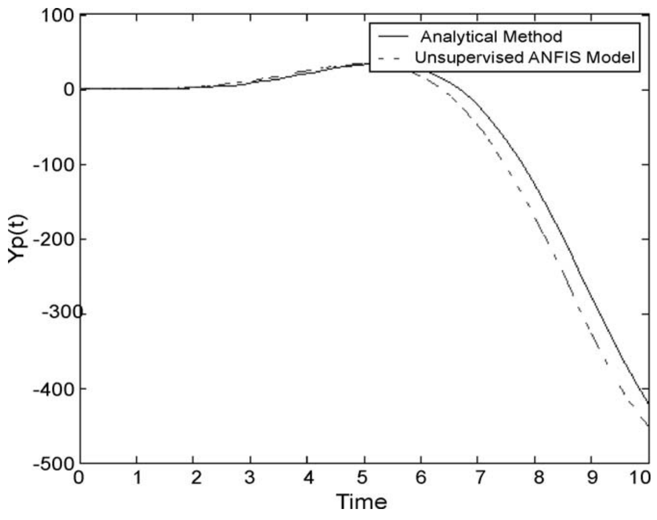

Fig. 8. Analytical and the proposed methods' responses collation in Example 3.

algorithm combining the least square method and the back propagation (BP) algorithm is used to train the ANFIS. Besides, according to Fig. 2, training the ANFIS is started after calculation of $\mathbf{u_k}$ or in other words by finishing the iteration. However, during the assessment of our method we trained the ANFIS and created y_p by each iteration. Our aim was to compare the result of algorithm to the analytic answer by each iteration and observing the convergence of the algorithm.

4. Experimental results

We evaluated the accuracy of proposed approach by applying it to solve several differential equations. In this section some examples are presented and the solutions of our algorithm are compared with the real ones.

Example 1. First-order differential equation with constant excitation.

$$\frac{d}{dt}y(t) + 2y(t) = 1, \quad y(0) = 1 \tag{24}$$

The related trial function will be in the form of (25).

$$y_p(t) = 1 + (t - 0)u_p(y_p, W, B) \tag{25}$$

This solution satisfies the initial condition. After substitution of (25) in (24), desired response of unsupervised ANFIS (u_p) is found.

$$u_p(y_p, W, B) = \frac{-1 - t(d/dt)u_p(y_p, W, B)}{1 + 2t} \tag{26}$$

The solution of analytical and the proposed methods are compared in Fig. 3. Fig. 4 shows u, the response of ANFIS model

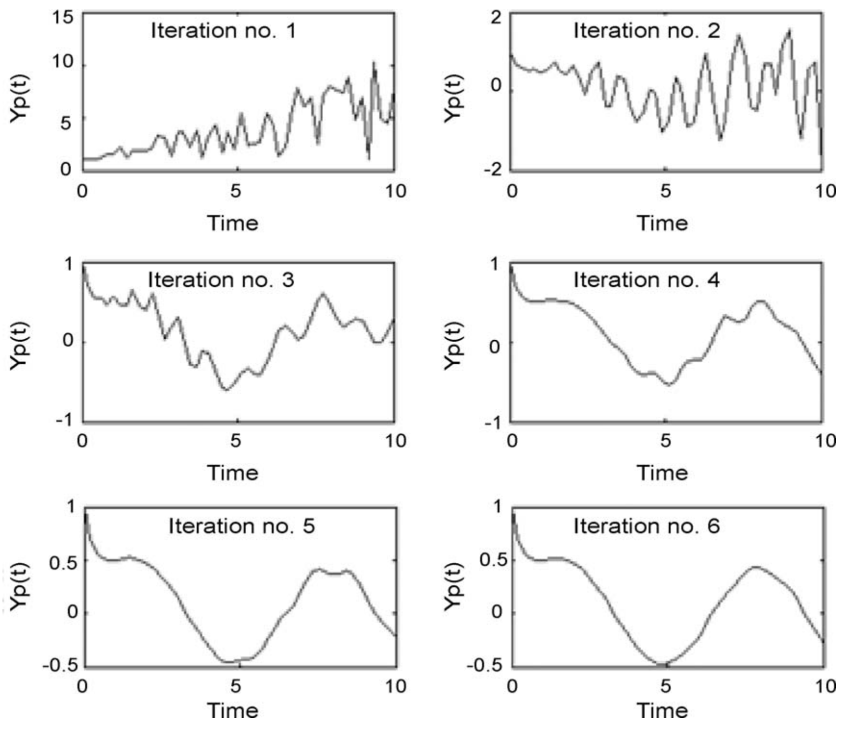

Fig. 7. Response of proposed algorithm in different iterations in Example 2.

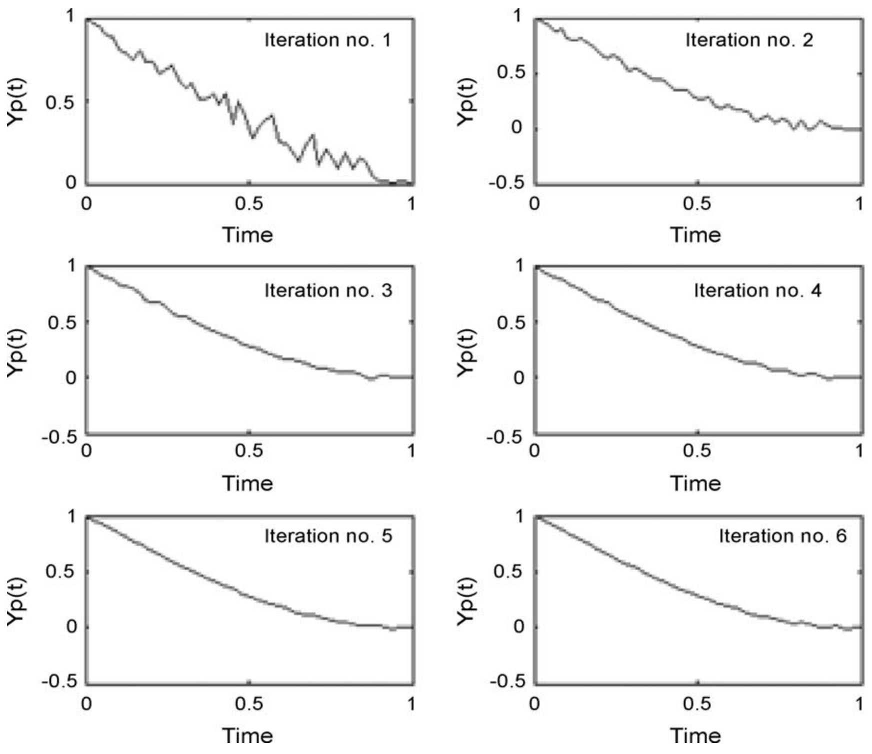

Fig. 9. Response of unsupervised ANFIS method in different iterations in Example 4.

after convergence. The output of the proposed algorithm for 6 sequential iterations (1–6) is depicted in Fig. 5. As can be seen in this figure ANFIS output is random in the first iteration, because the input and desired output are random. But after the fourth iteration, ANFIS output approaches the real value.

Example 2. First-order differential equation with sinusoidal excitation.

$$\frac{d}{dt}y(t) + 2y(t) = \sin(t), \quad y(0) = 1 \tag{27}$$

Desired response of unsupervised ANFIS (u_p) is found in (28).

$$u_p(y_p, W, B) = \frac{\sin(t) - t(d/dt)u_p(y_p, W, B) - 2}{1 + 2t} \tag{28}$$

The responses of analytical and the proposed methods are compared in Fig. 6. The convergence procedure of the proposed approach is shown in Fig. 7.

Example 3. First-order differential equation with nonlinear sinusoidal excitation.

$$\frac{d}{dt}y(t) + 2y(t) = t^3 \sin\left(\frac{1}{2}t\right), \quad y(0) = 1 \tag{29}$$

The responses of analytical and proposed methods are compared in Fig. 8. In this case, our algorithm converges by just one iteration of training which demonstrates its training speed.

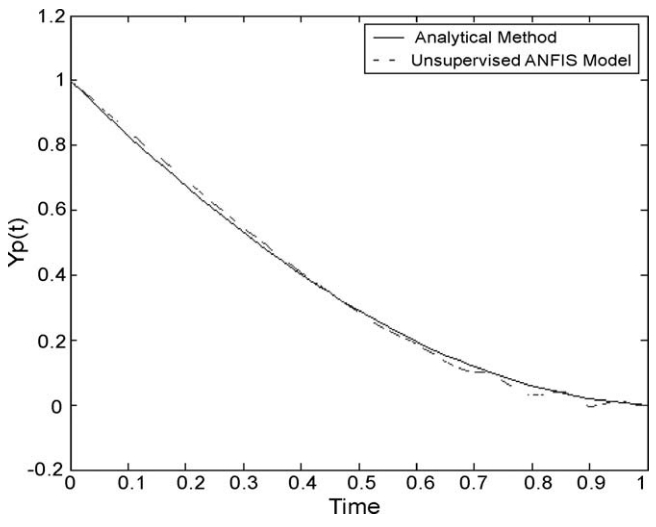

Fig. 10. Comparison of analytical and proposed methods in Example 4.

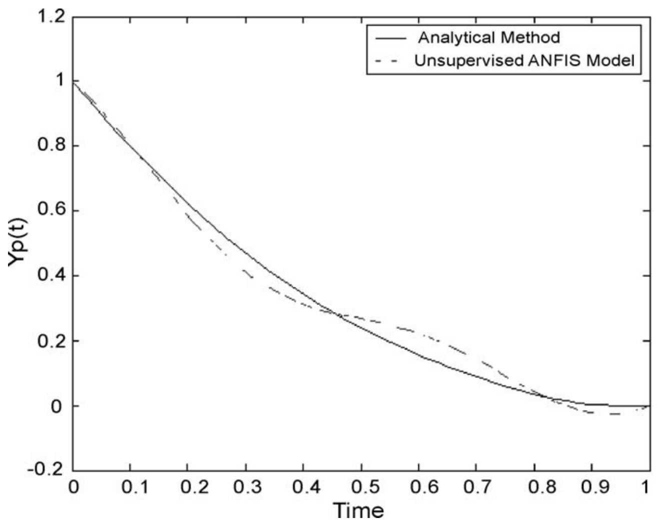

Fig. 11. Comparison of analytical and proposed methods in Example 5.

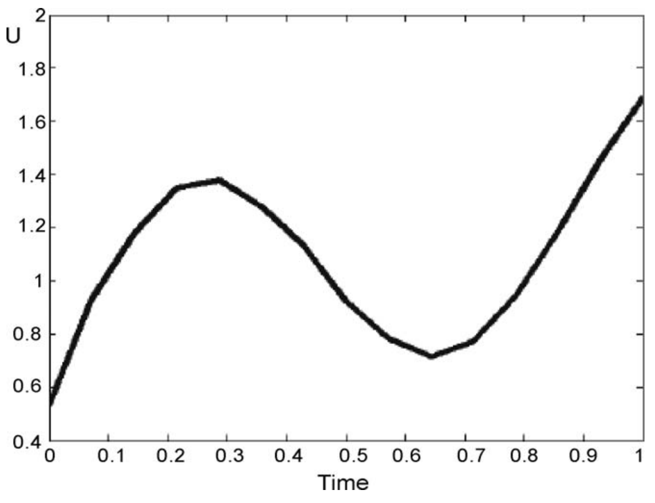

Fig. 12. ANFIS output after convergence for Example 5.

Example 4. Second-order differential equation with constant excitation.

$$\frac{d^2}{dt^2}y(t)+y(t)=2,\quad y(0)=1,\ y(1)=0 \tag{30}$$

The related trial function would be in the following form if $y(t_0)=y_0$, $y(t_1)=y_1$.

$$y_p(t)=\frac{t_1y_0+t_0y_1}{t_1-t_0}+\frac{y_1-y_0}{t_1-t_0}t+(t-t_0)(t-t_1)u_p(y_p,W,B) \tag{31}$$

This solution satisfies the boundary condition. After substitution of (31) in (30), desired response of unsupervised ANFIS (u_p) is found.

$$u_p(y_p,W,B)=\frac{\begin{array}{c}2-(1-t)-2(2t-1)(d/dt)u_p(y_p,W,B)\\ -t(t-1)(d^2/dt^2)u_p(y_p,W,B)\end{array}}{2+t(t-1)} \tag{32}$$

Fig. 9 shows the algorithm's output in different iterations and the acceptable response is achieved after second iteration. Also analytical response is compared with our answer in Fig. 10.

Example 5. Second-order differential equation with time-varying input signal. This example shows the case of variable-time input signal.

$$\frac{d^2}{dt^2}y(t)+y(t)=2+2\sin(4t)\cos(3t),\quad y(0)=1,\ y(1)=0 \tag{33}$$

Desired response of unsupervised ANFIS (u_p) is found in (34).

$$u_p(y_p,W,B)=\frac{\begin{array}{c}2+2\sin(4t)\cos(3t)-(1-t)-2(2t-1)\\ (d/dt)u_p(y_p,W,B)-t(t-1)(d^2/dt^2)u_p(y_p,W,B)\end{array}}{2+t(t-1)} \tag{34}$$

Analytical response is compared with our answer in Fig. 11. Also the achieved $u_p(y_p, W, B)$ is shown in Fig. 12.

5. Conclusion

This paper presented a novel approach for solving differential equations which utilizes an unsupervised ANFIS. The accuracy of the proposed method was examined by solving first-order and second-order differential equations with input excitation signal in both constant and time-varying formats. The achieved results demonstrate that the accuracy and fast convergence of the novel approach, which takes advantages of unsupervised ANFIS in its initial form, is comparable with the results of similar approaches that use neural networks. This is due to the capability of ANFIS to approximate stochastically unknown functions and relations; and also the trial solution which is in a close and differentiable form and satisfies the boundary/initial conditions.

In the future, we will develop our method to solve nonlinear and partial differential equations. If the capability of solving partial differential equation is added to this method, it would be easily extended to solve high-dimensional problems. Moreover, solving nonlinear differential equations enhances the generalization property of this method.

References

[1] A. Cichocki, R. Unbehauen, K. Weinzierl, R. Holzel, A new neural network for solving linear programming problems, European Journal of Operational Research 93 (1996) 244–256.
[2] A. Castro, V. Miranda, Mapping neural networks into rule sets and making their hidden knowledge explicit application to spatial load forecasting, in: Proceedings of the 14th Power System Computation Conference, 2002.
[3] S. Mitra, Y. Hayashi, Neuro-fuzzy rule generation: survey in soft computing framework, IEEE Transactions on Neural Networks 3 (2000) 748–768.
[4] J.-S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Transaction on System Man and Cybernet 23 (5) (1993) 665–685.
[5] A.I. Nuno, B. Arcay, J.M. Cotos, J. Varela, Optimization of fishing predictions by means of artificial neural networks, ANFIS, functional networks and remote sensing images, Expert Systems with Applications 29 (2005) 356–363.
[6] A. Noureldin, A. El-Shafie, M.R. Tahab, Optimizing neuro-fuzzy modules for data fusion of vehicular navigation systems using temporal cross-validation, Engineering Applications of Artificial Intelligence 20 (2007) 49–61.
[7] S.P. Singh, A.S. Raghuvanshic, Adaptive intelligent hydro turbine speed identification with water and random load disturbances, Engineering Applications of Artificial Intelligence 20 (6) (2007) 795–808.
[8] K.C. Lee, P. Gardner, Adaptive neuro-fuzzy inference system (ANFIS) digital predistorter for RF power amplifier linearization, IEEE Transactions on Vehicular Technology 55 (January (1)) (2006) 43–51.
[9] E.D. Ubeyli, I. Guler, Adaptive neuro-fuzzy inference system to compute quasi-TEM characteristic parameters of micro shield lines with practical cavity sidewall profiles, Neurocomputing 70 (2006) 296–304.
[10] H. Qin, S.X. Yang, Adaptive neuro-fuzzy inference systems based approach to nonlinear noise cancellation for images, Fuzzy Sets and Systems 158 (10) (2007) 1036–1063.
[11] P. Çivicioglu, Using uncorrupted neighborhoods of the pixels for impulsive noise suppression with ANFIS, IEEE Transactions on Image Processing (2007), doi:0.1109/TIP.2007.891067.
[12] G. Daoming, C. Jie, ANFIS for high-pressure water jet cleaning prediction, Surface & Coatings Technology 201 (2006) 1629–1634.
[13] A. Depari, A. Flammini, D. Marioli, A. Taroni, Application of an ANFIS algorithm to sensor data processing, IEEE Transactions on Instrumentation and Measurement 56 (February (1)) (2007) 75–79.
[14] K. Assaleh, Extraction of fetal electrocardiogram using adaptive neuro-fuzzy inference systems, IEEE Transactions on Biomedical Engineering 54 (January (1)) (2007) 59–68.
[15] M.-L. Huang, H.-Y. Chen, J.-J. Huang, Glaucoma detection using adaptive neuro-fuzzy inference system, Expert Systems with Applications 32 (2007) 458–468.
[16] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural network for solving ordinary and partial differential equations, IEEE Transactions on Neural Networks 9 (5) (1998) 987–1000.
[17] K. Hornik, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.
[18] M. Hüsken, C. Goerick, Fast learning for problem classes using knowledge based network initialization, in: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00), 2000.
[19] N. Smaoui, S. Al-Enezi, Modeling the dynamics of nonlinear partial differential equations using neural networks, Journal of Computational and Applied Mathematics 170 (1) (2004) 27–58.
[20] R. Brause, Adaptive modeling of biochemical pathways, in: Proceedings of the 15th IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI'03), 2003.
[21] S. Hea, K. Reif, R. Unbehauen, Multilayer neural networks for solving a class of partial differential equations, Neural Networks 13 (2000) 385–396.
[22] L. Manevitz, A. Bitar, D. Givoli, Neural network time series forecasting of finite-element mesh adaptation, Neurocomputing 63 (2005) 447–463.
[23] T. Leephakpreeda, Novel determination of differential-equation solutions: universal approximation method, Journal of Computational and Applied Mathematics 146 (2002) 443–457.
[24] A. Malek, R. Shekari Beidokhti, Numerical solution for high order differential equations using a hybrid neural network—optimization method, Applied Mathematics and Computation 183 (1) (2006) 260–271.

[25] N. Mai-Duy, T. Tran-Cong, Numerical solution of differential equations using multi quadric radial basis function networks, Neural Networks 14 (2001) 185–199.
[26] L. Jianyu, L. Siwei, Q. Yingjian, H. Yaping, Numerical solution of elliptic partial differential equation using radial basis function neural networks, Neural Networks 16 (2003) 729–734.
[27] D.R. Parisi, M.C. Mariani, M.A. Laborde, Solving differential equations with unsupervised neural networks, Chemical Engineering and Processing 42 (2003) 715–721.
[28] Jang J-SR, C.T. Sun, E. Mizutani, Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice-Hall, Englewood Cliffs, NJ, 1997.