Novel AI based On-Line Sequential Learning Technique for High Performance DC Servo motor Control

Vikas Kumar, Prerna Gaur, A.P. Mittal

Division of Instrumentation and Control Engineering, Netaji Subhas Institute of Technology, Sector-3, Dwarka, New Delhi
E-mail: sainivika@gmail.com , prernagaur@ yahoo.com, mittalap@gmail.com

Abstract: In this paper a neuro-fuzzy based adaptive tracking controller which is trained when the controller is operating in an online mode for high performance DC servo motor control is presented. The proposed structure consists of five layer feed-forward network which is trained using sequential learning method. Extreme Learning Machine (ELM), a recently developed novel method for the training of the single hidden layer feed forward neural networks (SLFNs) is used to initialize the training algorithm with a small chunk of training data. The membership function for each rule is determined using heuristics based methods and the consequent parameters of the Tagaki-Sugeno-Kang (TSK) type fuzzy inference are then determined in an online manner using the recursive least square method. The performance of the proposed technique in terms of the training time, training accuracy for tracking a reference trajectory is evaluated and is compared with the adaptive neuro-fuzzy based controller and other existing faster training algorithms such as ELM. The robustness of the proposed scheme is tested under DC motor parameters variations such as armature resistance, viscous friction and moment of inertia for all implemented controllers. Results obtained ensure the robustness of the proposed controller versus other implemented controllers.

Keywords: Artificial Neural Network, Back Propagation, Single Hidden Layer Feed forward networks, Sequential Learning, Recursive Least square, fuzzy clustering

1. INTRODUCTION

DC servo motors are small size and low inertia motors, normally used in precise control applications such as computer numerically controlled machines and robotics for positioning of the machine tools end effectors (Mccomb and predco, 2006). Among the various tracking control schemes available for the precise speed and/or position control of DC servo motor, the conventional PID control and Sliding Mode Control (SMC) are the most dominant as these controls schemes are simple and easy to implement. However these control techniques rely on the mathematical dynamics of the plant and cannot deal with the nonlinearity present in the process (Zumberge and Passino, 1996). In order to have a model free design and to meet the stringent performance requirements in terms of speed of response and accuracy, intelligent control techniques like fuzzy logic and artificial neural networks are developed (Passino and Yurkovich, 1998). However, a fuzzy logic controller for implementation needs a large number of parameters to be tuned by trial and error method if the high performance control is required. For the neural network controller implementation it is very difficult to generate the training data for all operating modes of the plant. Combining fuzzy logic and neural network, a neuro-fuzzy controller can expand the reasoning capabilities of fuzzy logic and the learning capabilities of the neural network and is applied successfully by researchers and academicians in applications ranging from classification problems to the adaptive control problems. The training time for ANN and/or neuro-fuzzy based controllers can be up to several hours or days, because ANN or a neuro-fuzzy controller when trained with traditional gradient based methods generally trains very slowly as the error has to back propagate recursively and also the training can converge to local minima. A lot of research work has been reported in literature for the fast and efficient training of FFNN (Leung et al., 2001) and (Wong and Leung, 2006).

Recently Huang *et. al.* has proved that a single hidden layer feed-forward neural network (SLFN) with nonlinear activation function in the hidden layer and linear activation function in the output layer can approximate any non linear function (Huang et al., 2006) and developed a novel method of training SLFNs called Extreme Learning Machine (ELM). In ELM the input weights connecting the input nodes and hidden nodes are assumed randomly and the weights connecting the hidden layer and output layer are then analytically determined using the Moore-Penrose generalized inverse. It is observed that ELM makes the learning speed hundreds to thousands times faster than the traditional learning algorithm (Liu and Wang, 2010) and can never converge to local minima. ELM has been successfully applied for applications like sensor less speed control of PMSM drive (Kumaret et al., 2013). An adaptive-network-based fuzzy inference system (ANFIS) is an example of batch learning fuzzy inference system which can learn data batch by batch basis. The training requires cycling of the entire data over a number of epochs which can be time consuming, the no. of epochs, learning rate and no. of rules needs to be determined prior to the training. For applications where data is available on one by one basis, many sequential learning methods like evolving Tagaki-Sugeno and Sequential Adaptive fuzzy Inference System) (Angelov and Filev, 2005), self-constructing neural fuzzy inference network (SONFIN) are used (Juand and Lin, 1998), which can learn one by one basis. Similar to the above methods Rubaai et. al. used an adaptive learning of neural networks for speed control of DC motor. The proposed technique has shown the significant improvement over traditional gradient based methods, but the algorithm needs the learning rate to be determined prior to the learning (Rubaai and Kotaru, 2001) which is a difficult task as the learning rate has significant influence on the stability of the convergence algorithm. Bhushan *et. al.* has made a comparative study of the ANN and its variants viz. multilayer perceptron, Elman and radial basis function network based control schemes (Bhushan et al., 2012). In their research work the implemented algorithms performed satisfactorily, but the implemented controllers are trained offline and adaptation to the DC machine parameter variations remains unaddressed.

In order to overcome the ANN training problems like tuning a large number of parameters prior to the training, generating a large number of training data, large training time, the convergence of a training algorithm to a local minima and adaptation to the process parameter variations, a novel on-line sequential learning based neuro-fuzzy controller (OS-NF) for the trajectory control of the DC servo motor is implemented in this work. The proposed control technique requires almost no parameters to be determined prior to the training, is free from convergence to local minima and can be trained in an online manner in order to adapt to the process parameter changes. The robustness of the proposed control scheme is investigated under unknown DC machine parameter variations. The performance of the proposed controller is compared with the (1) regular neuro-fuzzy (Adaptive neuro-fuzzy based controller) in terms of tracking error and training time (2) ELM and ANN based controllers in terms of training time so as to identify and to establish the robustness of the proposed control scheme.

This paper is organised as follows: Section 2 deals with the mathematical modelling of the implemented control techniques. In section 3 mathematical modelling of DC motor as a discrete time nonlinear system is presented. Simulink implementation details are presented in section 4. The results of the implemented controllers are presented and a comparative study of the performance of these controllers is presented in section 5 conclusions are presented in section 5.

2. ARTIFICIAL NEURAL NETWORKS

The schematic Single hidden layer Feed forward networks (SLFNs) are discussed in this section.

2.1 Single hidden layer feed forward networks (SLFNs)

The structure of single hidden layer feed forward network is as shown in Fig. 1.
For N distinct input-output data samples (P_i, Y_i)
$P_i = [P_{i1}, P_{i2}, \ldots P_{iN}]^T \epsilon R^n$,
$Y_i = [Y_{i1}, Y_{i2}, \ldots Y_{iN}]^T \epsilon R^n$
The output of the hidden layer (V) with $\check{N}$ hidden nodes and activation function $\Phi(x)$ can be expressed as:

$$V = \sum_{i=1}^{\check{N}} \Phi(w_i P_j + \theta_i) \quad (1)$$

j=1, 2…N, θ_i =Bias of i^th hidden neuron, $w_i = [w_{i1}, w_{i2}, \ldots w_{in}]^T$ is the weight vector connecting the i^th hidden node and the input node.
The overall output of the SLFN can be expressed using (2)

$$\sum_{i=1}^{\check{N}} \gamma_i \ \Phi(w_i P_j + \theta_i) = O_j; j=1,2...,N \quad (2)$$

$\gamma_i = [\gamma_{i1}, \gamma_{i2}, \ldots \gamma_{i\check{N}}]^T$ is the weight vector connecting the i^th hidden node and the output node.
The network shown in fig. 1 with n number of hidden nodes can be trained to learn n distinct samples with zero error, i.e. $\sum_{j=1}^{\check{N}} || o_j - Y_j || = 0$ i.e. there exist γ_i, w_i and θ_i, such that (2) holds true (Liu and Wang, 2010)
The training of the above network is equivalent to minimize the cost function ,

$$J = \sum_{j=1}^{N} \left(\sum_{i=1}^{\check{N}} \gamma_i g(w_i P_j + \theta_i) - t_j \right)^2 \quad (3)$$

Traditionally the gradient based methods are used to adjust w_i, γ_i and θ_i recursively, so that so that cost function in (3) is minimized.

2.2 ELM Algorithm

An ELM algorithm is a faster and efficient training method of SLFNs is based upon the following two theorems: The detailed proof of the theorems has not been given here as it is rigorously proved by Huang et al., 2006.

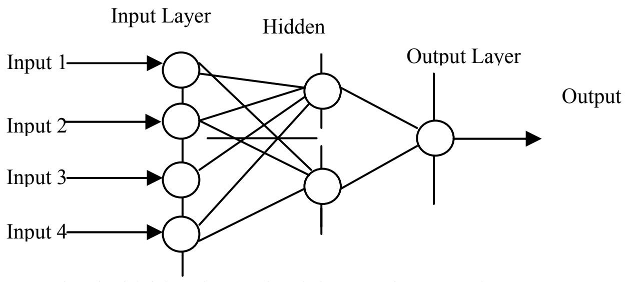

Fig.1. Single hidden layer feed forward network.

Theorem 1. Given a standard SLFN with $\check{N}$ hidden nodes and activation function Φ which is infinitely differentiable in any interval, for N arbitrary distinct samples(P_i, Y_i) when $P_i \epsilon R^n$ and $Y_i \epsilon R^m$, fon any w_i, θ_i randomly chosen from the interval R^n and R, according to the continuous, then with probability one the hidden layer output matrix V is invertible with $\|V\gamma - Y\| = 0$.

Theorem 2. Given any small positive value $\epsilon > 0$, and activation function Φ: R→R which is infinitely differentiable in any interval, there exists $\check{N} \leq N$, such that for N arbitrary distinct samples (P_i, Y_i) where $P_i \in R^n$ and $Y_i \in R^m$ for any w_i, θ_i randomly chosen from the interval R^n and R respectively, according to any continuous probability distribution, then with probability one $\|V_{Nx\check{N}} * \gamma_{\check{N}xm} - Y_{Nxm}\| < \epsilon$

The two theorems can be summarized as: given a SLFN with number of training samples equals the number of hidden nodes, there is no need to tune weight and biases of the hidden nodes. They can be randomly assumed and the output weight matrix can be calculated by inverting the hidden layer matrix in a single step. This results in a zero training error. When the number of hidden nodes is less than the number of training samples $\check{N} \leq N$, one can still randomly assigns parameters of the hidden nodes and the output weight matrix is calculated by inverting the hidden layer matrix with a small training error.

Based upon the two theorems explained above the ELM algorithm is summarized as follows:
Step 1: randomly assign input weights w_i and bias θ_i for i = 1... Ň
Step 2: Calculate the hidden layer output matrix V
Step 3: Calculate the output weight matrix γ, using $\gamma = V^{\dagger}T$, where $V^{\dagger}$ is Moore-Penrose generalized inverse of matrix V.
In step 3, $\gamma = V^{\dagger}T$ is one of the least squares solutions of a general linear system or has the smallest norm among all the least-squares solutions of $V\gamma = Y$. The matrix V can be a non-square or a singular matrix and hence Moore-Penrose generalized inverse is used to implement step 3. The calculation details of the Moore-Penrose inverse can be found in appendix.

As the output weight matrix is determined analytically using Moore-Penrose generalized inverse, the training time for ELM is mainly spent on the calculation of Moore-Penrose inverse. The training speed can be 100 to 1000 times faster than the gradient based methods where error has to back propagate recursively so as to minimize the training error.

2.3 Adaptive Neuro-fuzzy controller

The adaptive Neuro fuzzy controller is a five layer feed forward network as shown in fig 2 in which each node performs a particular function on incoming signals depending upon the function assigned to the respective node. The controller has two states, a learning state and a controlling state. In the learning state, the performance evaluation is carried out according to the feedback which represents the process state. The ANFIS based controller is trained offline using gradient based methods. In the controlling state the controller is required to perform the desired objective. First order Takagi-Sugeno and Kang (TSK) based inference mechanism is used. The function of each layer is explained below:

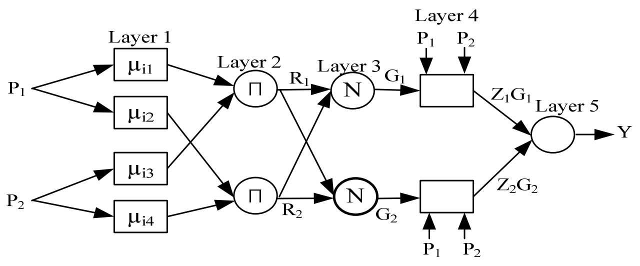

Fig. 2. Structure of regular neuro-fuzzy controller.

Layer 1: Each node in this layer is an input node and no processing is done at this layer.
Layer 2: is a Membership layer. Each node in this layer corresponds to the one linguistic label of the input node. For Gaussian type membership function the output of layer 2 is calculated using

$$\mu_i(P_i) = \exp\left[-(P - c_i)^2/(\sigma_i^2)\right] \tag{4}$$

c_i and σ_i are the centre and variance of the membership function for ith input variable.
Layer 3: is a rule layer. Each node in this layer is a fuzzy rule. The output of this layer is calculated by the product operation. The strength of each rule is calculated by dividing the firing strength of individual rule with the summation of firing strength of all rules. The firing strength of individual rules is calculated using logical AND operator as

$$R_i(P; C_i, \sigma_i) = \mu_{a_{1i}}(P_1; C_{1i}, \sigma_i)\ AND\ \mu_{a_{2i}}(P_2; C_{2i}, \sigma_i) AND \ldots \mu_{a_{ni}}(P_n; C_{ni}, \sigma_i) \tag{5}$$

For L number of rules, the firing strength of individual rules is normalized using (6)

$$G(P, C_i, \sigma_i) = \frac{R_i(P, C_i, \sigma_i)}{\sum_{i=1}^{L} R_i(P, C_i, \sigma_i)} \tag{6}$$

Layer 4 and 5: This layer is known as consequent layer. This system output of TSK fuzzy inference system is calculated using (7), the input to this layer is the output of the layer 3 and the consequent parameter matrix is Z_i.

$$Y_i = \sum_{i=1}^{L} Z_i G(P, C_i, \sigma_i) \tag{7}$$

2.4 On-line learning based Neuro-Fuzzy Control Structure (OS-NF)

The structure of the proposed control scheme is shown in fig. 3. The implementation of OS-NF is based on the functional equivalence of TSK type fuzzy inference system with the SLFNs by (Rong et. al., 2009). The output of the regular neuro-fuzzy based as calculated in (7) is equivalent to the output of SLFNs. The Gaussian type membership function as described by (4) is taken as activation function of SLFN $\Phi(P_i)$, The algorithm is summarized as below:
Step 1: Normalization: Incoming process data is normalized in the range (-1, 1), of the process input and output. The normalization is done using

$$P_{ji} = \frac{P_{ji}}{\max\{abs(P_{ji})\}},\ i=1, 2\ldots N \text{ and } j=1, 2\ldots \check{N} \tag{8}$$

Step 2: The normalized data is partitioned using fuzzy based online clustering. The number of data partitions is equal to the number of hidden neurons or number of rules. The each data point in the data space is related to all the clusters via membership degree, and the summation of the clustered membership to all the cluster centres is assumed to be one. The clusters centre is assumed randomly and then from the input vector or the incoming data point using (9) the membership degree is calculated. The Gaussian type activation function is used in the first hidden layer

$$\Phi(P_i) = \exp\left[-(P_i - C_i)^2/({\sigma_i}^2)\right. \tag{9}$$

Where P_i is the input data, $c_i$is the cluster centre,
Step 3: The output of the membership layer is normalized to obtain the firing strength of each rule using (5) and (6), this is equivalent to the output of the first hidden layer (V) as given in (1). The normalized output after fuzzification layer is given by (10)

$$y = \sum_{i=1}^{L} \gamma_i \Phi_i(P; C_i, \sigma_i) \tag{10}$$

The first order TSK model is specified by if-then rules which are proved to be a universal approximator. The rules for the first order TSK model is given by:

$$R_i: if\ P_1 is\ A_{i1}, AND\ P_2 is\ \ P_{i2}\ AND, \ldots AND\ \ P_M is\ A_{iM}, \text{Then}$$
$$Y_i = Z_{i0} + Z_{i1}P_{i1} + Z_{i2}P_{i2} + \cdots Z_{iM}P_{iM} \tag{11}$$

here $P_1, P_2 \ldots\ldots P_M$ are the fuzzy sets of the input variables, $A_{ji}(j = 1,2, \ldots M, i = 1,2, \ldots N)$ are the fuzzy sets of the jth input variable in the rule i and N is the dimension of input vector.

From (11), the consequent parameters of TSK FIS are the linear equation of input variables, and can be expressed as:

$$\gamma_i = P_{je}^T Z_i \tag{12}$$

Z_i is the consequent parameter matrix and P_{je}^T is extended input vector as defined in (13)

$$P_{je}^T = [1\ P_j^T]^{\ T} \tag{13}$$

The overall output of the TSK model for L rules is obtained by summing the output of each individual rule

$$y_j = \sum_{i=1}^{L} P_{je}^T Z_i \Phi_i(P, c_i, \alpha_i) = Y_j \tag{14}$$

Equation (14) is equivalent to the output of a TSK type regular neuro fuzzy controller as calculated in (7), (14) Can be rewritten as:

$$VZ = Y, \tag{15}$$

Initially for small training data, calculate the initial hidden layer matrix V_0, initial consequent parameter matrix,

$$Z_0 = P_0 V_0^T Y_0, \tag{16}$$

$P_0 = (V_0^T V_0)^{-1}$, and Y_0 is the initial chunk of output data samples.

Step 4: In this step the parameters of TSK type fuzzy inference are determined using sequential recursive least square algorithm. The sequential recursive least square algorithm is presented in the following section:

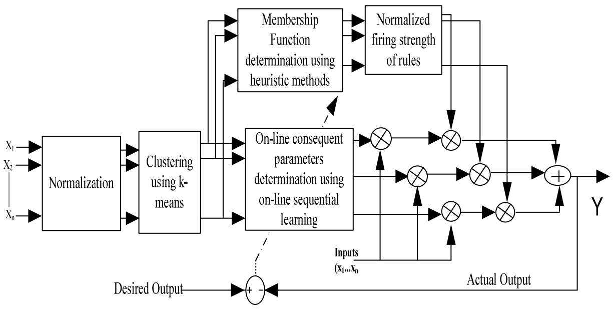
Membership Function determination using heuristic methods
Normalized firing strength of rules
X_1
X_2
X_n
Normalization
Clustering using k-means
On-line consequent parameters determination using on-line sequential learning
Inputs (x_1...x_n
Y
Desired Output
Actual Output

Fig. 3. Structure of OS-NF based controller.

For the incoming data sample, calculate the hidden layer matrix, V_{k+1}, using (1). Adjust the consequent parameter matrix using following update equations (Rong et al., 2009):

$$P_{k+1} = P_k - P_k V_{k+1}^T (I + V_{k+1} P_k V_{k+1}^T)^{-1} V_{k+1} P_k\ V_{k+1}^T \tag{17}$$

$$Z_{k+1} = Z_k + P_k V_{k+1}^T (Y_{k+1} - V_{k+1} Z_k) \tag{18}$$

The algorithm continues and the consequent parameters are updates till there exists error between the desired trajectory and the output trajectory as per equation (17) and (18).

3. DISCRETE TIME NONLINEAR SYSTEM

Mathematically the dynamics of the DC motor in discrete time domain is presented in the following section (Gopal, 2003).

$$v_a(t) = R_a i_a(t) + L_a \frac{di_a}{dt} + e_b(t) \tag{19}$$

$$e_b(t) = K_b \omega(t) \tag{20}$$

$$T_M(t) = K_T\ i_a\ = J \frac{d\omega(t)}{dt} + \ B\omega(t) + T_L(t) + T_F \tag{21}$$

$v_a(t) =$ applied armature voltage, $e_b(t) =$ back emf (volts), $i_a(t)$= armature currents (amps), R_a =armature winding resistance, L_a=armature winding inductance, $\omega(t)$=angular velocity of the motor rotor (rad/sec), $T_M(t)$=torque developed by the motor, K_T=torque constant, K_b=back emf constant, J=moment of inertia of the motor rotor with attached mechanical load, B= viscous friction constant of the motor rotor with attached mechanical load, $T_L(t)$=disturbance load torque, T_F = Frictional torque.

The load torque can be expressed as

$$T_L(t) = \ \Psi(\omega) \tag{22}$$

$\Psi(.)$ depends on the nature of the load. For most propeller driven or fan type loads the function $\Psi(.)$ takes the following form

$$T_L(t) = \ \mu\omega^2(t)[sgn\omega(t)] \tag{23}$$

DC motor drive system can be expressed as single input single output system by combining equations (18)-(23)

$$L_a J \frac{d^2\omega(t)}{dt^2} + (R_a J + L_a B) \frac{d\omega(t)}{dt} + (R_a B + K_b K_T)\omega(t) +$$
$$L_a \frac{dT_L(t)}{dt} \ + R_a[T_L(t) + T_F] + K_T v_a(t) = 0 \tag{24}$$

The discrete time model is derived from (24) as:

$$L_aJ\left[\frac{\omega(k+1)-2\omega(k)+\omega(k-1)}{T^2}\right]+(R_aJ+L_aB)\left[\frac{\omega(k+1)-\omega(k)}{T}\right]+(R_aB+K_bK_T)\omega(k)+L_a\frac{T_L(k)-T_L(k-1)}{T}+R_aT_L(k)+R_aT_F+K_Tv_a(k)\quad=0 \qquad (25)$$

$$T_L(k) = \mu\omega^2(k)[sgn\omega(k)] \qquad (26)$$

$$T_L(k-1) = \mu\omega^2(k-1)[sgn\omega(k-1)] \qquad (27)$$

$$\omega(k) \triangleq \omega(t=kT);\ \text{k=0,1,...} \qquad (28)$$

Where T= sampling period. Using (27), (25) can be rewritten as:

$$\omega(k+1)= K_1\omega(k)+K_2\omega(k-1)+K_3[sgn\omega(k)]\omega^2(k)+K_4[sgn\omega(k)]\omega^2(k-1)+K_5v_a(k)+K_6 \qquad (29)$$

$$K_1=\frac{2L_aJ+T(R_aJ+L_aB)-T^2(R_aB+K_bK_T)}{L_aJ+T(R_aJ+L_aB)}$$
$$K_2=-\frac{L_aJ}{L_aJ+T(R_aJ+L_aB)}$$
$$K_3=-\frac{T(\mu L_a+\mu R_aT)}{L_aJ+T(R_aJ+L_aB)}$$
$$K_4=\frac{T\mu L_a}{L_aJ+T(R_aJ+L_aB)}$$
$$K_5=\frac{K_TT^2}{L_aJ+T(R_aJ+L_aB)}$$
$$K_6=\frac{T_FR_aT^2}{L_aJ+T(R_aJ+L_aB)}$$

Following numerical values are the parameters of 1 HP, 220V, 550rad/s DC motor and are used in the simulation $J=0.068\ kg\ m^2, B=0.03475\ N.m/(rad/sec), R_a=7.56\Omega, L_a=0.055H, \mu=0.0039N.m/(rad/\text{sec})^2\ K_b=3.475volts/(rad/sec), K_T=3.475Nm/amp, T_F=3.475\ Nm, T=0.04$ sec

Using above parameters for DC motor the numerical values of the constants are as follows:
$K_1=0.03466,\ \ K_2=-0.1534069, K_3=-2.286928e-3,$
$K_4=3.5193368e-4, K_5=0.2280595, K_6=-0.105284$

(19)- (29) are used to model the DC motor with propeller or fan driven load. The line diagram of the DC servo motor is shown in Fig. 4.
In order to implement the above mentioned controllers, control law is derived using the discrete time modelling equations of the DC motor in the following section.
From the mathematical modelling of the DC motor the output speed of the motor is expressed using (29) as

$$\omega(k+1)=f\big(\omega(k),v_a(k)\big) \qquad (30)$$
Similarly,

$$\omega(k+2)=f\big(\omega(k+1),v_a(k+1)\big) \qquad (31)$$

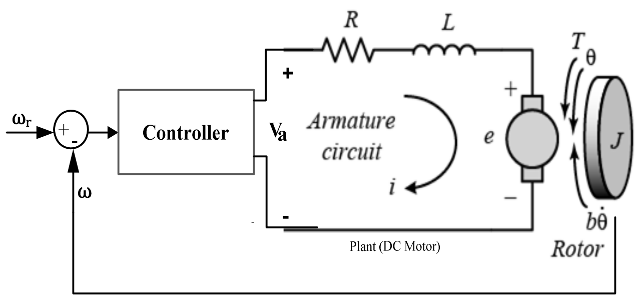

Fig. 4. Line diagram for the DC Servo Drive.

Using (30), (31) can be rewritten as:

$$\omega(k+2)=f\left(f\big(\omega(k),v_a(k)\big),v_a(k+1)\right)\ \text{ or}$$
$$\omega(k+2)=F\big(w(k+1),v_a(k+1)\big) \qquad (32)$$

For nth order plant, we can rewrite (32) as

$$\omega(k+n)=f(\omega(k+n-1),v_a(k+n-1)) \qquad (33)$$

The value of $v_a(k)$ is adjusted by the control algorithm so that the DC motor tracks desired trajectory. The control action can be expressed as a function of

$$v_a(k)=G(\omega(k),\omega(k-n)) \qquad (34)$$

where n=0,1,2…k-1. The objective of the learning is to minimize the following cost function as given in (35)

$$J(k)=\sum||\omega_r(k+1)-f(f(w(k),v_a(k))||^2 \qquad (35)$$

$$\omega_r\ (k)=10*\sin\left(2*pi*k*\frac{T}{4}\right)+16*\sin\left(2*pi*k*\frac{T}{7}\right)$$
(36)

T is the sampling time. For the present work T is taken as 0.004s.

4. SIMULINK IMPLEMENTATION

The DC servo motor is simulated in MATLAB/SIMULINK. The proposed controllers are implemented so that the DC motor follows the reference trajectory given in (36) in the presence of unknown parameter variations.

4.1 ANN based controller

The ANN based controller has feed forward architecture as shown in fig 5. The training data is generated using (29) for random values of armature voltage.
SLFN is made to learn the following function:

$$v_a(k+1)=G(\omega(k),\omega(k-1),\omega_r(k+1))) \qquad (37)$$

So that the following error function is minimized.

$$J_{ANN,ELM}(k) = ||\omega_r(k+1)-NN(\omega(k),\omega\big(k-1),vak||2 \qquad (38)$$

In order to minimize the error function in (38), the weights and biases of the ANN inverse model are recursively updated using

$$w_k = w_{k-1} - \eta \frac{\partial e(k)}{\partial w} \quad (39)$$

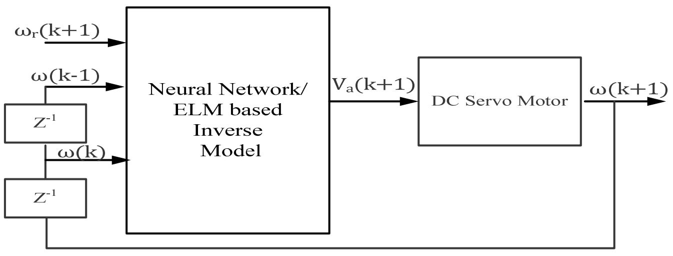

Fig. 5. ANN/ELM based control scheme.

In the present work the number of hidden neurons is taken as 20, the activation function of the hidden layer is 'tansig' and that of the output layer is 'purlin'. The network is trained for 1000 no. of epochs. The training time is 10s and the training accuracy achieved is 2.5*e-3.

4.2 ELM based controller

ELM based controller as shown in fig. 5 is trained to learn the inverse dynamics of the plant by minimizing the cost function as given in (38). The weights of the output layer are analytically determined using Moore-Penrose generalized inverse. The number of hidden neurons is 20, activation functions in the hidden layer is *tansig* and output layer is *purelin*, are taken to be the same as in case of ANN based controller. The weight and biases of the hidden layer neurons are randomly chosen and the weights of the output layer are determined analytically. The training time taken for the ELM based controller is 0.157s and the training accuracy achieved is 6.04e-4.

4.3 Regular Neuro-Fuzzy based Controller

The line diagram for regular neuro-fuzzy type controller is shown in fig. 6. The number of rules is 20 is chosen to be same as in case of on-line sequential learning based neuro-fuzzy controller.

The network is trained offline with hybrid training method (i.e. back propagation and recursive least square). The control input is

$$v_a(k+1) = f(e(k), \frac{de(k)}{dt}) \quad (40)$$

And is determined using regular neuro fuzzy controller by minimizing the following cost function given by (41)

$$J_{NF,OS-NF}(k) = ||\omega_r(k+1) - \omega(k)||^2 \quad (41)$$

The training time for neuro-fuzzy based controller is 4.275s and the training accuracy achieved is 4.82e-5.

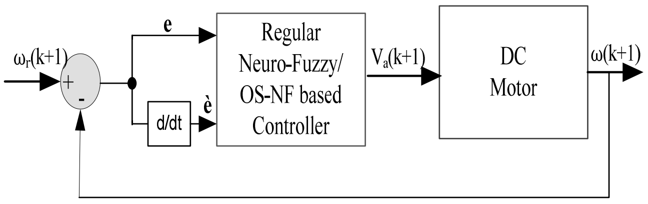

Fig. 6. Regular neuro-fuzzy/OS-NF based control scheme.

4.4 On –line Sequential learning based Neuro-Fuzzy (OS-NF) controller

The line diagram OS-NF controller is shown in fig 6. The control objective is same as explained in section 4.3. Instead of assuming the parameters of the membership function randomly (Rong et al., 2009) we have normalized the incoming data between (-1, 1) based upon the plant knowledge, the membership functions are then determined using heuristic based methods as explained in section 2.4. The training is initialized with the small chunk of training data, the no. of samples being equal to the number of rules. Now the training data is presented to the network on one by one basis and this training data is discarded once the training completes. For a set of 20 rules the training time was 0.7432s and the training accuracy achieved is 2.379e-7.

To study the robustness of the implemented controllers for machine parameter variations, the armature resistance, moment of inertia and viscous friction is increased by 100% of their original value. The Mean of Square Error (MSE) is calculated for each implemented controller and the results are compiled in tabular form.

5. RESULTS AND DISCUSSIONS

The trajectory of the motor is controlled using ANN based controller, ELM based controller, regular neuro-fuzzy based controller and on-line sequential learning based neuro-fuzzy controller. The response of the ANN based controller for the reference trajectory is shown in fig. 7, the blue line shows the reference trajectory and the red line shows the controlled trajectory. The MSE for ANN based controller is found out to be 0.2157 and the training time is 19s. For the trajectory control, ELM based controller offered better tracking response shown in fig. 8 as compared to the ANN based controller shown in fig. 7. The MSE for the ELM based controller is found to be 0.1959 and the training time 0.157s, is drastically reduced compared with the ANN based controller

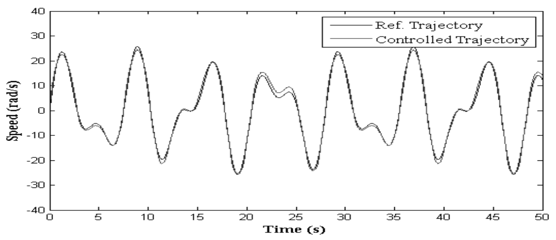

Fig. 7. Response of ANN based controller to a reference trajectory.

For regular neuro-fuzzy, the trajectory control response is shown in fig. 9. The response of the regular neuro-fuzzy based controller is better as compared to the ELM based controller, as the MSE for regular neuro-fuzzy based controller is 0.1892. The training time for regular neuro-fuzzy controller (4.275s) is large as compared with the ELM based controller but is small as compared to ANN based controller. The OS-NF based controller offered the best tracking response with a MSE of 0.1728 and the training time 0.7432s is very small as compared with ANN and regular neuro-fuzzy based controller.

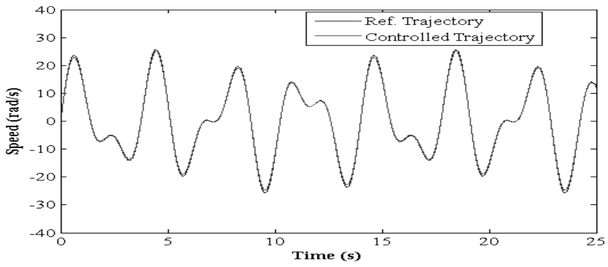

Fig. 8. Response of ELM based controller to a reference trajectory.

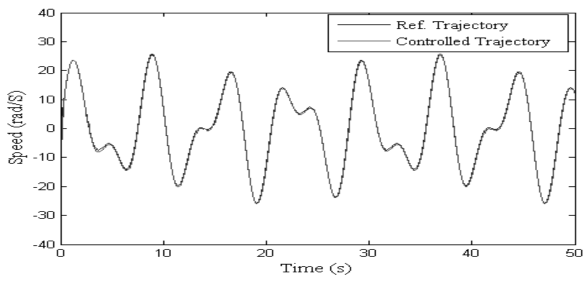

Fig. 9. Response of regular neuro-fuzzy based controller to a reference trajectory.

The response of OS-NF controller is shown in fig. 10. The on-line sequential learning based neuro-fuzzy controller The performance comparison of the implemented controllers is shown in fig. 11, in which the black line shows the response of OS-NF controller which is found to be the best among the implemented controllers as it has offered the best tracking response with a MSE of 0.1728 and the training time is 0.7432 seconds.

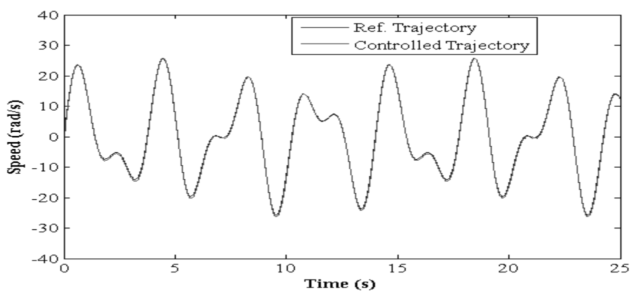

Fig. 10. Response of OS-NF controller to a reference trajectory.

5.1 Effect of Parameter variations

The parameters of the DC motor like armature resistance, moment of inertia and viscous friction are varied from 0% to 100% of their original value. Two cases are considered for the performance evaluation. In the first case the armature resistance is increased by 100% and in the second case the viscous friction and the moment of inertia are increased by 100% and the results obtained for all implemented controllers are compared.

5.1.1 Effect of Armature Resistance

The armature resistance is increased by 100%. The performance comparison of all implemented controllers for 100% armature resistance change is shown in fig. 12.

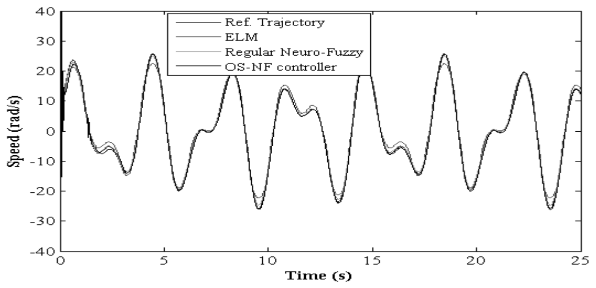

Fig. 11. Performance comparison of implemented controllers.

The performance of ANN based controller deteriorates due to the armature resistance change when compared to unchanged armature resistance as shown in fig. 7. The MSE is increased by 323% from 0.2157 to 0.9132 for ANN based controller. The response of the ELM based controller is also affected due to armature resistance change as MSE is increased by 315% from 0.1959 to 0.8131. ELM similar to ANN based controller is trained offline. For regular neuro-fuzzy controller the MSE is increased by 245% from 0.1892 to 0.6543. The OS-NF controller is able to suppress the effects of armature resistance in an effective way the MSE for OS-NF controller is increased by 171% from 0.1728 to 0.4693. The percentage change in MSE for a 100 % increase in armature resistance (171%) is least for OS-NF controller as compared to all other implemented controllers. The response of the OS-NF controller seems to be the least affected and the response of the ANN based controller is the most affected due to the armature resistance change as shown in fig. 12.

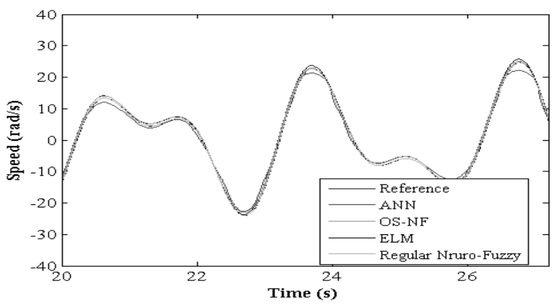

Fig. 12. Comparison of implemented controllers (Ra is increased by 100%).

5.1.2 Effect of Moment of Inertia and Viscous Friction

In the second case, the moment of inertia and viscous friction coefficient are increased by 100% of their initial value. The performance comparison of all implemented controllers for 100 % increase in the moment of inertia and viscous friction is shown in fig. 13. The MSE for ANN based controller is increased by 207% from 0.2157 to 0.6638. The ELM based controller offers a slightly better response compared to ANN based controller. The MSE for ELM based controller increased by 134% from 0.1959 to 0.4549. The MSE for regular neuro-fuzzy based controller is increased by 118% from 0.1892 to 0.4139 and MSE for OS-NF controller is increased by 49% from 0.1728 to 0.2587.

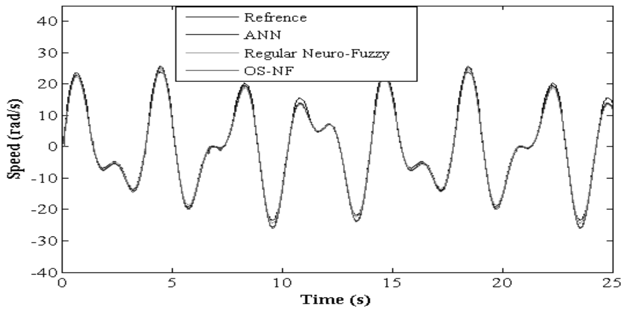

Fig. 13. Comparison of implemented controllers (B and J are increased by 100%).

The % change in MSE for a 100 % increase in the moment of inertia and viscous friction (49%) is least for OS-NF controller. The summary of the results is given in table 1

6. CONCLUSION

The performance of online sequential learning based neuro-fuzzy controller is compared with ANN based controller, ELM based controller and regular neuro-fuzzy controller for trajectory tracking of the DC servo motor under parameter variations. The performance of all the implemented control strategies is evaluated taking the mean of square error (MSE), training time and training accuracy achieved as the performance criteria.

The highest training accuracy is achieved for on-line sequential learning based neuro-fuzzy controller compared to the rest of the implemented control strategies. The training time for the proposed scheme is also very small. The performance of the proposed control scheme is robust to the machine parameter variation, as the % change in MSE is least for both cases i.e. up to 100 % armature resistance change and 100% change in the moment of inertia and viscous friction. On-line sequential learning based neuro-fuzzy controller with its on-line learning capability, very small training time and very high training accuracy can be effectively applied in the real time applications

APPENDIX

In this section the Moore Penrose generalized inverse is introduced. For a SLFN if the output neuron contains linear activation function, then a linear relation exists between the hidden layer matrix, output weight matrix and final output of the network. For SLFN $VZ = Y$, the Z matrix can be calculated by inverting matrix V, but matrix V can be a non singular matrix or a non square matrix. Hence Moore-Penrose inverse is used to calculate minimum norm least square solution.

Moore Penrose Inverse: a matrix G is said to be a Moore-Penrose matrix of A if it satisfies:
$AGA = A, GAG = G, (AG)^T = AG, (GA)^T = GA$

Table 1. Performance comparison of implemented controllers.

Controller ↓ Type	Training Time(s)	Training Accuracy	MSE	MSE for Ra Changed	% Increase in MSE for Ra Change	MSE for B and J Changed	% Increase in MSE for B & J Change
ANN	19	2.51e-3	0.2157	0.9132	323	0.6638	207
ELM	0.157	6.04e-6	0.1959	0.8131	315	0.4549	134
Regular Neuro-Fuzzy	4.275	3.8e-5	0.1892	0.6543	309	0.4139	118
OS-NF Controller	0.7432	2.7e-6	0.1728	0.4693	177	0.2587	49

REFERENCES

Angelov, P., and Filev, D., (2005). “Simpl_eTS: A simplified method for learning evolving Takagi–Sugeno fuzzy models,” in *Proc. 14th IEEE Int. Conf. Fuzzy Syst.*, pp. 1068–1073.

Bhushan, B., Singh, M. and Hage, Y., (2012). “Identification and control using MLP, Elman, NARXSP and radial basis function networks: a comparative analysis,” *Artif. Intell Rev Vol. 37*, pp. 133-156.

Gopal, M. (2003).Digital Control and State Variables Methods. *McGraw-Hill*, New Delhi, India.

Huang, G.B., Zhu, Q.Y. and Siew, C.K., (2006). “Real Time Learning Capability of Neural Networks,” *IEEE Transactions Neural Networks, vol. 17, no. 4*, pp. 863-878.

Huang, G.B., Zhu, Q.Y. and Siew, C.K., (2006). “Extreme learning machine: Theory and applications,” *Neurocomputing vol. 70*, pp. 489–501.

Juang, C-F., Lin, C-T., (1998). “An On-Line Self-Constructing Neural Fuzzy Inference Network and Its Applications,” *IEEE Transactions on Fuzzy Systems*, Vol. 6, No. 1, pp. 12-32.

Jang, J.-S. R. (1993). “ANFIS: Adaptive-network-based fuzzy inference system,” *IEEE Trans. Syst., Man, Cybern.*, vol.23, no.3, pp. 665–685.

Kumar, V., Gaur, P., Mittal, A.P. Singh. B., (2013). “ELM based Sensor less Speed Control of PMSM Drive,”. *IJVAS Vol.-11 No. 2/3*, pp. 190-204.

Leung, C.S.,Tsoi, A. C. and Chan, L.W. (2001).“Two Regularizers for Recursive Least Squared Algorithms in Feed forward Multilayered Neural Networks,”. *IEEE Transactions Neural Networks, vol. 12, no. 6*, pp. 1313–1332.

Liu, P.X., Meng, M., and Hu, C. (2005). “On-Line Data-Driven Fuzzy Clustering With Applications To Real-Time Robotic Tracking,”. In *Procedings of the 4 International Conference on Robotics & Automation*, New Orleans, LA, pp. 5039-5044.

Liu, N., and Wang, H. (2010). Ensemble Based Extreme Learning Machine. *IEEE Signal Processing Letters vol. 17, no. 8*, pp. 754-757.

Mccomb, G., and Predko, M.(2006). Robot Builder’s Bonanza. *McGraw-Hill*, Columbus, USA.

Passino, K.M., and Yurkovich, S.(1998). Fuzzy Control. *Addison Wesley Longman, Menlo Park*, CA.

Rong, H.J., Huang, G.B., Sundararajan, N. and Saratchandran, P. (2009). “Online Sequential Fuzzy Extreme Learning Machine for Function Approximation and Classification Problems,” *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, vol. 39, no. 4*, pp 1067-1072.

Rubaai, A., and Kotaru, R. (2000). “Online Identification and Control of a DC Motor Using Learning Adaptation of Neural Networks,”. *IEEE Trans. Industry Applications, Vol. 36*, pp. 935-942.

Uddin, M.N., and Wen, H. (2007). “Development of a Self-Tuned Neuro-Fuzzy Controller for Induction Motor Drives,” *IEEE Transactions on Industry Applications, Vol. 43*, pp 1108-116.

Xu, Y., Wong, K.W. and Leung, C.S. (2006). “Generalized RLS approach to the training of neural networks,”. *IEEE Trans. Neural Networks. Vol. 17, No. 6*, pp.19–34.

Zumberge, J., and Passino, K.M. (1996). “A case study in intelligent vs. conventional control for a process control experiment”. in *Procedings of the International Symposium on intelligent control*, Dearborn, Michigan, pp. 37 – 42.