RTRL Algorithm Based Adaptive Controller for Non-linear Multivariable Systems

K.C. Sindhu Thampatty
IEEE Student Member
Dept. of Electrical Engineering
National Institute of Technology
Calicut, Kerala

M. P. Nandakumar
Dept. of Electrical Engineering
National Institute of Technology
Calicut, Kerala

Elizabeth P. Cheriyan
Dept. of Electrical Engineering
National Institute of Technology
Calicut, Kerala

ABSTRACT
The paper presents a new design of adaptive and dynamic neural network-based controller architecture with feedback connection for non-linear multivariable systems. The network is trained on-line at each sampling interval using the desired output trajectory and the training method used is the Real Time Recurrent Learning Algorithm (RTRL). The recurrent network is a fully connected one, with feedback from output layer to the input layer through a delay element. Since the synaptic weights to the neurons are adjusted on-line, this controller has potential applications in real time control also. Moreover, it can be used for both continuous and discrete systems. The simulation results obtained by applying the algorithm to a non-linear multivariable system demonstrate the effectiveness of the proposed method.

Keywords
Artificial Neural Network (ANN), Non-linear Control, Multivariable System, Real Time Recurrent Learning Algorithm (RTRL).

1. INTRODUCTION
Adaptive controllers in general are designed for dynamic systems having unknown parameters, under the assumption that these parameters are linear, which is unrealistic for complex non-linear systems. Since most of the commonly encountered time-varying systems are non-linear in nature, the recent advances in non-linear control theory inspired the development of adaptive control schemes for these systems. One of the successful areas of application of neural networks is non-linear control system because neural networks are capable of producing highly non-linear mappings.

Most of the results reported in the literature of the adaptive control of non-linear dynamical systems are related to single-input single-output systems (SISO), while, the practical systems have multiple inputs and multiple outputs (MIMO). Hence, our interest in this work is to develop a controller for multivariable systems. In such systems, unlike in SISO systems, there exists more than one control loop which will cause loop interactions. These interactions can cause system instability or result in poor control performance. A multivariable controller can achieve non-interacting controls, but the design methods involve the construction of a mathematical model describing the dynamics of the plant. In practice, the exact model representation of the plant is difficult to obtain. A complex system requires an intelligent controller with adaptive and learning capabilities in the presence of unknown disturbances, unmodeled dynamics and unstructured uncertainties.

The ability of Artificial Neural Network (ANN) to model non-linear systems can be readily exploited in the synthesis of non-linear controllers. The neural networks can learn sufficiently accurate model and can give good non-linear control when model equations are not known or when only partial state information is available. Hence, neural network-based controller design is an efficient approach to process non-linearity as well as variable interactions.

In control applications, neural networks can be incorporated in direct strategy or indirect strategy [5]. In direct strategy, for a given current state of the system and the target state for the next sampling instant, the network is trained to produce the control action that drives the system to the target state. In indirect strategy, the network is trained with input-output data from the dynamic system. For a given current state and current control action, the network learns to predict the next state of the system. In this work, a direct control strategy has been adopted to design the controller.

A number of researchers proposed methods for identification and control of non-linear multivariable systems. In complex control system applications, adaptive and robust control techniques have high potential, because the modification of controller parameters are based on convergence and stability constraints, which can limit the performance of adaptive systems. AL-Zahary *et.al* [1] explains the realization of neuro controllers for non-linear multivariable systems. They suggested two extended models of SISO neurocontroller to realize multivariable systems for identification and control, in which the controls are generated by training the unknown models with available input-output data. Different control strategies [6], [8], [11], [16] and [19] .using neural network with different training methods like feed forward architecture with back propagation learning algorithm [3] are available in literature A methodology to develop a proper model for the design of a robust controller for multivariable system is explained in [17], where the controller is designed for structurally ill-conditioned processes. Different aspects of controller design for non-linear systems are available in literatures [4] and [20].

In most of the training techniques, retaining the information about the infinite past are not possible, which is a drawback for the real-time applications. In feed forward neural networks, back propagation with adaptive learning rate is the most widely used gradient based algorithm. But this algorithm is very sensitive to

noise and relatively unstable. Hence there is a need for a real-time recurrent learning algorithm, which stabilizes the system and also improves the convergence. A suitable architecture for this type of learning is a fully connected dynamic recurrent neuron architecture in which the output of the neuron is fed back to the input through a delay. Different training methods are available for recurrent neurons [8], [13] and [18]. A method for non-linear system control using ANN is suggested in paper [15], but that is applicable only for SISO systems.

In all these reported works, fully connected recurrent neural network is seldom used for the real time control of non-linear multivariable systems. Thus an intelligent controller is required to replace the conventional controllers. Among the several architectures found in literature, recurrent neural networks (RNN) involving dynamic elements and internal feedback connections have been considered as more suitable for real time applications. The critical issue in the application of recurrent neural network is the choice of the network architecture, i.e., the number and type of neurons, the location of feedback loops and the development of a suitable training algorithm. In this paper we discuss the use of RTRL algorithm to train the network, which is an optimal algorithm such that it minimizes the instantaneous squared error at the output neuron for every discrete time instants. Number of neurons in the output layer is equal to the number of states of the system and it is a fully connected recurrent neural network, i.e., all the outputs from the neuron is fed back to the input through a delay.

2. REAL TIME RECURRENT LEARNING ALGORITHM

This is a forward gradient algorithm, which makes use of a matrix of partial derivatives of the network state values with respect to every weight. The algorithm is based on minimizing the instantaneous squared error at the output of the neuron. The main difficulty related to the recursive training of recurrent network arises from the fact that the output of the network and its partial derivatives with respect to the weights depend on the inputs. In this method, partial derivatives of each node with respect to each weight are computed at every iteration [14]. The method is completely online and is simple to implement.

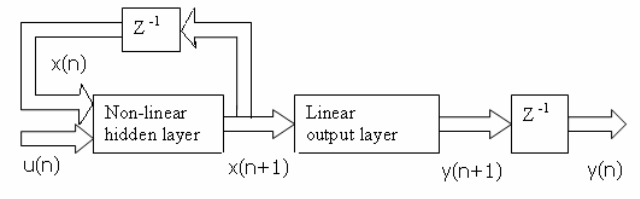


Figure 1. Layout of a fully connected recurrent network

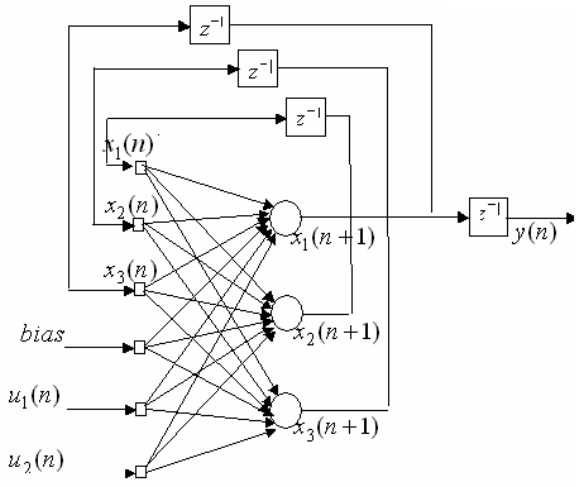


Figure 2. Structure of a fully connected recurrent network

Figure 1 shows the layout of a fully connected recurrent network and Fig. 2 shows its architecture [14] with three states, one bias and two control inputs. All the outputs are fed back to the source nodes through a delay. There are two distinct layers for the network, a *concatenated input feedback layer* and a *processing layer* of computational nodes.

If q is the number of states of the system and m is the number of control inputs to the system, then the dynamic system can be represented by the non-linear difference equation given in (1) and (2).

$$\boldsymbol{x}(\boldsymbol{n}+1) = \phi[\boldsymbol{W}_a \boldsymbol{x}(\boldsymbol{n}) + \boldsymbol{W}_b \boldsymbol{u}(\boldsymbol{n})] \tag{1}$$

$$\boldsymbol{y}(\boldsymbol{n}) = \boldsymbol{C}\boldsymbol{x}(\boldsymbol{n}) \tag{2}$$

Where $\boldsymbol{W}_a$ is a $q \times q$ matrix, $\boldsymbol{W}_b$ is a $q \times (q+m+1)$ matrix and $\boldsymbol{C}$ is a $p \times q$ matrix. Let ψ: R $\rightarrow$R is a non-linear map. The neural network with m inputs, p outputs and q states can be represented in state space form as given in (3), (4) and (5).

$$\boldsymbol{x}(\boldsymbol{n}+1) = \begin{bmatrix} \phi(\boldsymbol{W}_1^{\,T}\zeta(\boldsymbol{n})) \\ \cdots\cdots\cdots \\ \phi(\boldsymbol{W}_j^{\,T}\zeta(\boldsymbol{n})) \\ \cdots\cdots\cdots \\ \cdots\cdots\cdots \\ \phi(\boldsymbol{W}_q^{\,T}\zeta(\boldsymbol{n})) \end{bmatrix} \tag{3}$$

$$W_j = \begin{bmatrix} W_{aj} \\ W_{bj} \end{bmatrix} \quad (4)$$

$$\zeta(n) = \begin{bmatrix} x(n) \\ u(n) \end{bmatrix} \quad (5)$$

The matrices W_a, W_b, C and the non-linear function ϕ are interpreted as follows:

- In (1), the total weights are split into two, namely, W_a, and W_b. The matrix W_a represents the synaptic weights associated with the q neurons in the hidden layer that are fed back as inputs in the input layer. The matrix W_b represents the synaptic weights associated with this hidden layer, which are connected to the input sources including the bias. Thus, the bias terms of the hidden neurons are included in W_b.
- The matrix C represents the synaptic weights of p output neurons connected to the hidden layers. The neurons in the hidden layers are with hyperbolic tangent non-linear function given by :

$$\phi(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \quad (6)$$

Or the logistic function:

$$\phi(x) = \frac{1}{1 + e^{-2x}} \quad (7)$$

The network consists of a set of N fully connected neurons and a set of M inputs. Let $W_{ki}(n)$ denote the weight associated with the link originating from neuron i towards neuron k at time n.

The net input to neuron k, $S_k(n)$ is defined as the weighted sum of all activations in the network. Based on standard RTRL terminology, the output of node k at time $(n+1)$ is to be calculated as:

$$y_k(n+1) = f_k(S_k(n)) \quad (8)$$

$$S_k(n) = \sum_{i \in N \cup M} W_{ki} Z_i(n) \quad (9)$$

$$Z_k(n) = \begin{cases} x_k(n), & k \in M \\ y_k(n), & k \in N \end{cases} \quad (10)$$

The nonlinear activation function $f(.)$ maps to the range $[0,1]$ The overall network error at time n is defined by an error function $J(n)$ represented as:

$$J(n) = -\frac{1}{2} \sum_{k \in outputs} [y_k(n) - d_k(n)]^2 = -\frac{1}{2} \sum_{k \in outputs} [e_k(n)]^2 \quad (11)$$

where $d_k(n)$ denotes the desired target value for output k at time n.

To execute the RTRL algorithm, three matrices $\Lambda_j(n)$, $\bigcup_j(n)$ and $\Phi(n)$ are calculated which are explained in (12), (13) and (15)

1. $\Lambda_j(n)$, is a q x $(q+m+1)$ matrix defined as the partial derivative of the state vector with respect to the weight vector.

$$\Lambda_j(n) = \frac{\partial x(n)}{\partial w_j} \; ; \quad j = 1,2,.....q \quad (12)$$

2. $\bigcup_j(n)$) is a q x $(q+m+1)$ matrix whose rows are all zero except for the jth row, which is equal to the transpose of the vector ζ (n).

$$\bigcup_j(n) = \begin{bmatrix} O \\ \zeta^T(n) \\ O \end{bmatrix} ; \quad j = 1,2,.....q \quad (13)$$

3. $\Phi(n)$ is a q x q diagonal matrix whose diagonal elements are the partial derivatives of the non-linear activation function given by:

$$\phi(w_j^T(\zeta(n))) \; ; \; j = 1,2,.....q \quad (14)$$

$$\Phi(n) = diag(\phi^T(W_1^T(\zeta(n))),.....,\phi^T(W_q^T(\zeta(n))) \quad (15)$$

Proceeding through the steps of the LMS algorithm using Steepest descent method, the correction in the synaptic weights can be calculated as:

$$\Lambda_j(n+1) = \Phi(n)[Wa(n)\Lambda_j(n) + \bigcup_j(n)] \; ; \; j = 1,2,.....q$$

(16)

$$e(n) = d(n) - Cx(n) \tag{17}$$

where $e(n)$ is the error and $d(n)$ is the desired output in n^{th} instant. The correction in weight is given by:

$$\Delta W_j(n) = \eta C \Lambda_j(n) e(n) \tag{18}$$

Thus, the RTRL algorithm can be summarized as follows:

- Initialize the weights:

$$W_j = \begin{bmatrix} W_{aj} \\ W_{bj} \end{bmatrix}$$

Figure.3. Structure of the proposed controller

- Set the initial states as $x(0) = x_0$, where x_0 is calculated for any operating condition.
- Set $\Lambda_j(n)$ =0 for j = 1, 2, · · · , q
- Compute for n = 0, 1, · · ·

$$\Lambda_j(n+1) = \Phi(n)[Wa(n)\Lambda_j(n) + \bigcup_j(n)] \; ; \; j = 1,2,....q$$

$$e(n) = d(n) - Cx(n)$$

$$\Delta W_j(n) = \eta C \Lambda_j(n) e(n)$$

By identifying the partial derivatives of the output function with respect to the weights as sensitivity elements, the sensitivity can be denoted as given in (19):

$$p_{ij}^{k}(n+1) = \frac{\partial(y_k(n+1))}{\partial(W_{ij}(n))} \tag{19}$$

where

$$y = C[x] + D[u] \tag{20}$$

In this algorithm, the storage requirements cannot be reduced [15] as they constitute a crucial component in the weight upgradation procedure.

3. PROPOSED CONTROLLER

The proposed controller has been designed using the states of the system as the input signals, which are fed back from the output neurons through a delay. The control strategy at any instant can be written as:

$$y(n+1) = F(y(n), u(n), bias) \tag{21}$$

Where

$y(n) = [y_1(n), y_2(n), ..., y_p(n)]$: ***output vector***

$u(n) = [u_1(n), u_2(n), ... , u_m(n)]$: ***control vector***

When the desired output trajectory is given at any instant, say (n + 1)th instant, then the required control input is generated by the controller with the available knowledge of the states in n^{th} instant. For the purpose of training the network, the desired trajectory can be obtained by any method.

The controller generates an appropriate control to achieve the desired state trajectory after on-line training of the network using RTRL algorithm. The proposed controller is given in Fig.3. The number of neurons in the controller is same as the number of control inputs to the system. Each controller neuron is connected to all the states in the input layer and given an external bias.

The system states at (n + 1)th instant can be calculated as:

$$x(n+1) = \phi[(W_a x(n) + W_b K(n))] \tag{22}$$

Where $K(n) = [bias \; ; control \; inputs \; (n)]$

The desired output at (n + 1)th instant is given by:

$$y(n+1) = Cx(n+1) \tag{23}$$

The control input generated by the controller in (n + 1)th instant is given by:

$$u(n+1) = \sigma\left(\sum_i W_{ui} x_i(n)\right) \tag{24}$$

where W_{ui} is the synaptic weights associated with the controller neuron, σ is the activation function used in the controller and ϕ is the activation function used in the RTRL algorithm.

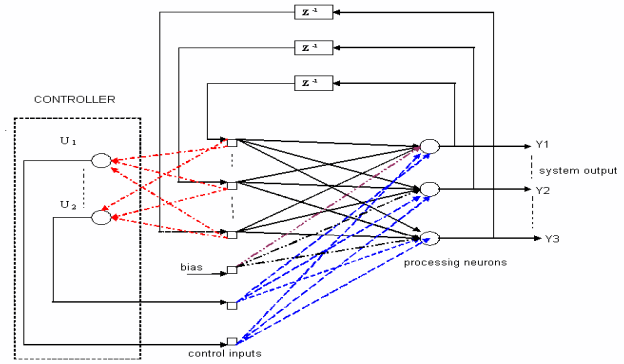


Figure.3. Structure of the proposed

4. (

In this section, two simulation studies one for SISO system and the second for MIMO system are conducted with the proposed adaptive control algorithm. For training the neural network, initially all the system state variables are assumed to a small value and given to the neural network. Then the desired system outputs

are given as the required closed loop state trajectories. These desired trajectories can be generated by any conventional methods. The synaptic weights of the neurons are first initialized to random values. After the desired outputs are presented to the network, at each instant, the error function $e(n)$ is computed and the correction in synaptic weights can be updated by using. (18). The value of learning rate parameter is problem dependent. In this work, we used a suitable value 0.5, which gives a fast convergence rate of the learning process. Hyperbolic tangent function given in .(6) is used as the activation function for the controller and the output neurons.

4.1 Single Input, Single Output systems (SISO)

The system considered here is an inverted pendulum problem, a classical problem in dynamics and control theory, widely used as benchmark for testing control algorithms. The objective of the controller is to control the angle of an inverted pendulum. The inverted pendulum has its mass above its pivot point and is mounted on a motor driven cart. In the system formulation, it is assumed that the pendulum moves only in a two dimensional space. The cart is driven by a motor that exerts a horizontal force F on the cart, which is the control input to the system. In this problem, balancing the pendulum in a vertical position in open loop is unstable. The proposed controller in feedback will modify the states of the system by generating a control input *u* to the system. The main objective of the controller in this system is to keep the pendulum upright in the presence of any disturbance by applying a control force *u*. The non-linear system equations of the inverted pendulum can be expressed as:

$$l\ddot{\theta} = mgl\sin(\theta) - ml^2\ddot{\theta} - ml\,\ddot{x}\cos(\theta) \quad (25)$$

$$M\ddot{x} = F - m(\ddot{x} + l\ddot{\theta}\cos(\theta) - l\ddot{\theta}^2\sin(\theta)) - k\dot{x} \quad (26)$$

where M is the mass of the cart, m is the mass of the pendulum, F is the force applied, k is a constant and l is the length of the pendulum. The system state variables are given by:

$$x_1 = \theta$$

$$x_2 = \frac{d\theta}{dt} = \dot{\theta}$$

$$x_3 = x = \textit{position of the cart}$$

$$x_4 = \frac{dx}{dt} = \dot{x} = \textit{velocity of the cart}$$

and the output of the system is given by:

$$[y] = [\theta]$$

In this problem, the desired trajectory used for training the network is obtained by pole placement technique of the linearised model of the system having a small settling time, about 2 seconds and reasonable damping with damping factor ξ = 0.5. Figures 4 to 7 shows the state trajectories of the system and Fig.8 shows the performance of the NN controller. In this example, the tolerance level is set to 0.005. The output of the controller, i.e., the required control input to the system, generated by the controller for the desired output is shown in Fig. 9.

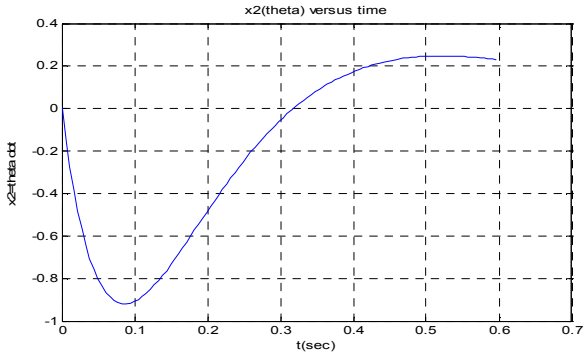


Figure 5. Angular velocity –Theta dot

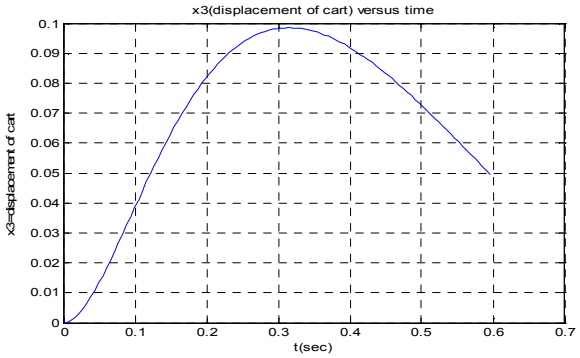


Figure 6. Position of the cart

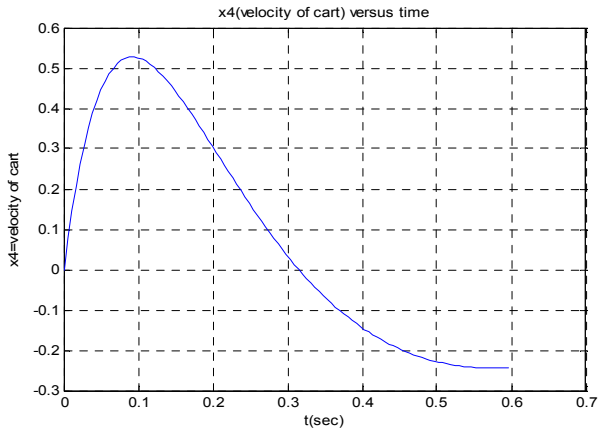


Figure 7 Velocity of the cart

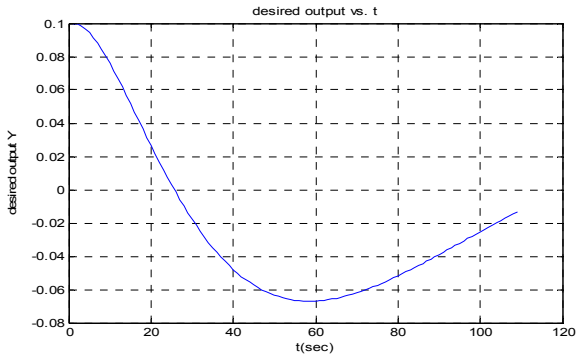
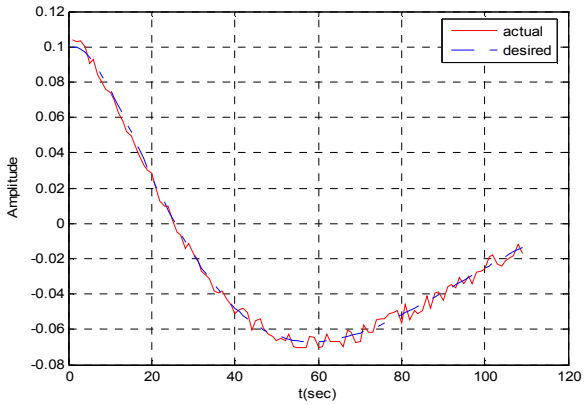


Figure 8. Adaptive NN controller performance



Once the desired trajectory as shown in Fig..8 is supplied to the network, the controller will act in such a way to track the output in very short time. The controller output is shown in Fig. 9 and the error square plot is shown in Fig. 10. The proposed controller is tested for different operating conditions (i.e., for different initial angles of the pendulum and the different initial positions of the cart) and it performs well, which shows its high robustness. The trained neural network can be used as the controller in the feedback circuit.

Figure 9. NN controller output

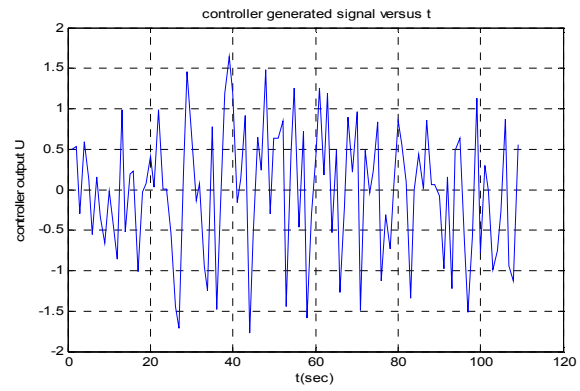
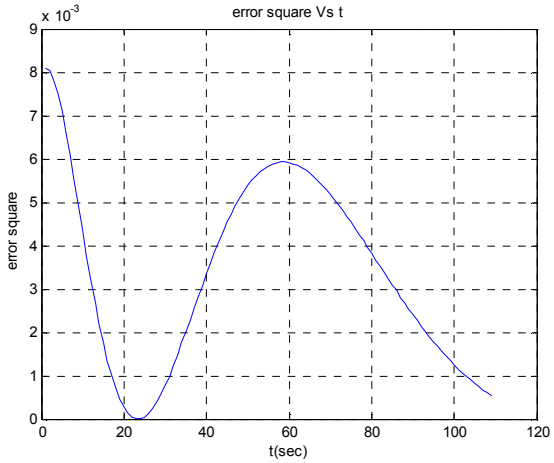


Figure 10. Squared Error



4.2 Multiple Input, Multiple Output Systems (MIMO)

In this example, we have considered a discrete model of a turbo-generator with six system states, two inputs and two outputs. The system dynamics are given as:

$$\frac{d\delta}{dt} = \boldsymbol{s} \tag{27}$$

$$\frac{d\boldsymbol{s}}{dt} = \boldsymbol{b}_1(\boldsymbol{P_T} - \boldsymbol{P_E}) \tag{28}$$

$$\frac{d\boldsymbol{E_q}}{dt} = \boldsymbol{b}_2(-\boldsymbol{Eq} + \boldsymbol{b}_3\boldsymbol{s}\sin(\delta - \alpha_{12}) + \boldsymbol{U}_1) \tag{29}$$

$$\frac{d\boldsymbol{P_T}}{dt} = \boldsymbol{b}_4(-\boldsymbol{P_T} + \boldsymbol{qC}) \tag{30}$$

$$\frac{d\boldsymbol{q}}{dt} = \boldsymbol{b}_6(-\gamma(\boldsymbol{q}) - \boldsymbol{b}_5\boldsymbol{s} + \boldsymbol{h}) \tag{31}$$

$$\frac{d\boldsymbol{h}}{dt} = \boldsymbol{b}_7(-\boldsymbol{h} + \boldsymbol{U}_2) \tag{32}$$

$$\frac{d\omega}{dt} = 0 \tag{33}$$

where $\boldsymbol{C}$, $\boldsymbol{b}$'s are constants. δ is the angle of rotation of the rotor, $\boldsymbol{s}$ is the slip, $\boldsymbol{P_T}$ is the mechanical power of the turbine, $\boldsymbol{P_E}$ is the electrical power generated and $\boldsymbol{Eq}$ is the generated

voltage. q is the valve movement of the regulator. U_1 and U_1 are the controls on voltage and valve position. α_{12} is the angle of transfer conductivity and h is the turbine velocity controller signal.

The objective of the controller is to make the deviation from the desired generated voltage to be zero and also the deviation of the generator load angular position to be zero. These are controlled by the throttle valve position and the loading torque of the generator. The desired trajectories for the outputs are supplied to the network. Then the network is trained in such a way that the mean squared error is minimized at the output. Figure 11 shows the desired trajectories of the outputs, i.e., the change in the desired voltage and the change in load angle, supplied to the network for training.

When these desired trajectories are supplied to the network, the control inputs to the system, U_1 and U_2 are generated by the controller in such a way that the minimum squared error is obtained in each sample.

In this case also the controller response is very fast. Figure 14 shows the efficiency of the proposed controller in tracking the outputs. Control inputs generated by the controller is shown in Fig.12 and the Error square graph of the network is shown in Fig.15 All these simulation results shows that the proposed controller is efficient in non-linear system control and since its response is very fast, this controller can be used for real time applications.

Figure.11. Desired trajectories of MIMO system

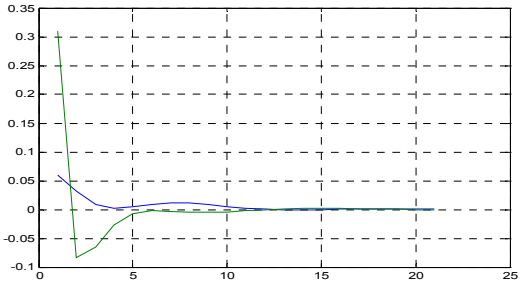


Figure.12. ANN Controller output

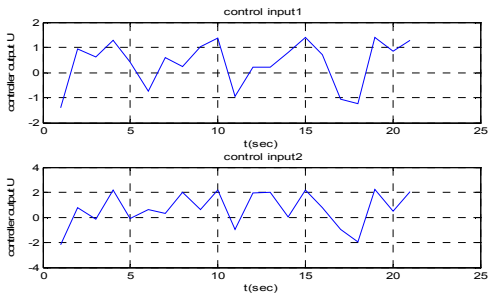


Figure.13. Adaptive NN Controller performance output1

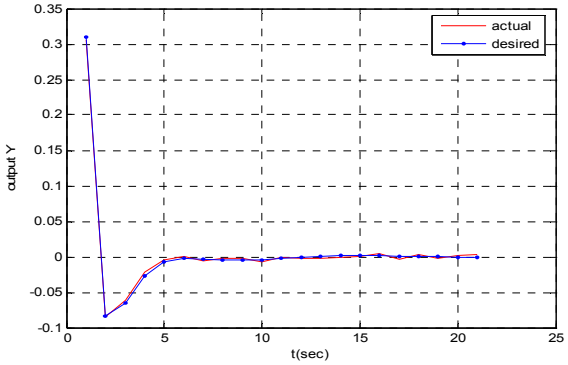


Figure.14. Adaptive NN Controller performance output2

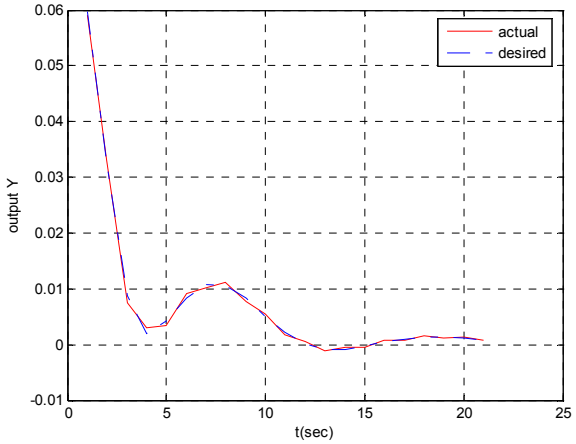
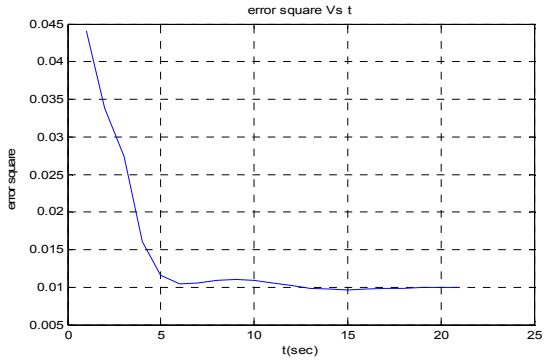


Figure.15. Squared Error



5. CONCLUSION

In this paper, we have described the implementation of an adaptive neural network-based controller for non-linear, multivariable, and dynamic systems. The fully connected dynamic neural network architecture selected for implementation has been trained by RTRL algorithm. The proposed controller is capable of generating the required control input by modifying the synaptic weights when the desired trajectory of output is presented to the

network. The quick response makes the ANN based approach very attractive for on-line applications in non-linear system control. The efficiency of the proposed controller has been demonstrated by applying the control strategy to both continuous and discrete systems, and the results agree with our claim.

REFERENCES

[1] Al-Zohary T.A, Wahdan, M. A. R. Ghonaimy and A. A. Elshamy.,2002 Adaptive Control of Nonlinear Multivariable Systems Using Neural Networks and Approximate Models,, Jounal of applied sciences

[2] Chang L.H., 2007. Design of nonlinear controller for bi-axial inverted pendulum, *IET Control theory Applications,* ,Vol 1, No.4, pp.979-986.

[3] Daniele Semino and Gabriele Pannocchia, 1999. Robust Multivariable Inverse-Based Controllers: *Theory and Application, Ind. Eng. Chem. Res*., Vol 38, pp.2375-2382.

[4] Duan G.R, Yu H.H., 2008. Robust pole assignment in high-order descriptor linear systems via proportional plus derivative state feedback, *IET Control theory Applications,* Vol 2, No.4, pp.277-287.

[5] Ender L , Maciel Filho R., 2000. Design of multivariable controller based on neural networks, *Elsevier, Computers and Chemical Engineering*, Vol 24, pp. 937-943.

[6] Filho. B, Cabral E, Soares A, 1998 . A new approach to artificial neural networks, *IEEE Trans on Neural Networks,* ,Vol.9, no. 6, pp.1167- 1179.

[7] Jagannathan Sarangapani, 1989. Neural Network Control of Nonlinear Discrete-Time, *Taylor and Francis ,New M. Young, The Technical Writers Handbook*, Mill Valley, CA: University Science,

[8] Jin. L, Nikiforuk. P, Gupta M., 1995. Approximation of discrete time state space trajectories using dynamic recurrent networks, IEEE Trans on Automatic Control, Vol.40, no.7, pp.1266-1270

[9] Linkens. D, Nyongesa Y, Learning systems in intelligent control: an appraisal of fuzzy, neural and genetic algorithm control applications, *IEE Proc. Control Theory App,* 134(4), pp 367-385

[10] Narendra K.S and Parthasarathy. K., 1990. Identification and control of dynamical system using neural networks, IEEE Trans on Neural Networks, Vol.1, no. 6, pp.4-27.

[11] Panos J. Antsaklis,. 1993. Neural Networks in Control Systems, *IEEE Trans on Energy Conversion,* Vol.8, no. 1, pp71-77.

[12] Paul J Webrose . An overview of Neural Networks for control, *IEEE Trans on Control Systems,* no. 1, pp40-42.

[13] Pearlmutter. B., 1995. Gradient calculations for Dynamic Recurrent Neural networks: A Survey, *IEEE Trans on Neural Networks*, Vol.6, no.3, pp 1212

[14] Simon Haykin, 1994. Neural Networks-A comprehensive Foundation , *IEEE Press*,

[15] Sindhu Thampatty K .C., M. P. Nandakumar, Elizabeth. P. Cheriyan, 2009. ANN based Adaptive Controller tuned by RTRL Algorithm for non-linear system control, *Proc. of Second International workshop on nonlinear Dynamics and Synchronisation, INDS'09*, July 20-21, Klagenfurt, Austria.

[16] Steepen W. Riche Webrose, 1994. Steepest Descent Algorithm for Neural Network controllers and Filters, *IEEE Trans on Neural Networks*, Vol.5, no. 2.

[17] Sun X.M, Zhao. J, Wang. W., 2008. State feedback control for discrete delay systems with controller failures based on average dwell-time method, *IET Control theory Applications*, Vol 2, No.2, pp.126-132.

[18] Williams R.J and Zipser D., 1989. A learning algorithm for continually Running fully Recurrent Neural Networks, *Neural Computation* , Vol 1, pp 552-558.

[19] Zhang. Y, Chen.G. P, Malik. O.P and Hope, G.S., 1992. An Artificial Neural Network Based Adaptive Power System Stabiliser, *IEEE Control System Magazine*, Vol.12, no. 4, pp8-10.

[20] Zhang .B., 2008. Parametric eigen structure assignment by state feedback in descriptor systems, *IET Control theory Applications*, Vol 2, No.4, pp.303-309.