

DESIGN AND DEVELOPMENT OF A NEURAL NETWORK (NN) CONTROLLER FOR A ROBOTIC ARM

Akaninyene M. Joshua Ph.D and Prof. Eneh I.I. and Ogbu, Mary N. C.

Dept of Electrical and Electronics Engineering, Enugu State of Science and Technology,
Enugu State.

ABSTRACT

Artificial Neural Networks (ANN) is an intelligent agent capable of being used in the control of nonlinear motions such as motions of a robot arm manipulator. ANN is capable of providing better control ability than traditional methods. The proposed controller has the ability to effectively utilize a large number of sensory information, can process data collectively and is adaptive by default. Using Back Propagation, the ANN is trained to imbibe the parameters of the robot arm manipulators for improved robot stability and suppressed vibration during robot operation. Mathematical models of the ANN are presented. Developed Simulink model is simulated and simulation result analyzed. Training performance result of 0.024 Root Mean Square Error (RSME) reduction at epoch 2 was achieved. The result show that trained network robot controller is capable of minimizing the system error to almost zero. A hybrid arrangement could be more responsive for better stability as robot arm manipulator controller.

Keyword: Artificial Neural Networks (ANN), Back Propagation, Simulink model, Root Mean Square Error (RSME).

1.0 Introduction

In many cases, when it is difficult to obtain a model structure for a system with conventional system identification techniques, the Artificial Neural Network (ANN), known as the intelligent techniques, is desired since it is capable of describing the system in the best possible way (Mersha, Stramigioli and Carloni, 2014) for further analyses and application possibilities. The artificial neural network systems are commonly used for modeling nonlinear dynamic systems. The main advantages of utilizing neural network for system identification are that they simultaneously evaluate many points in the parameter space and converge towards the global solution (Mersha, Stramigioli, Carloni, 2014). In contrast, neural network approaches for system identification offer many advantages over conventional ones especially in terms of flexibility and hardware realization (Erkaya, 2012). This technique is quite efficient in modeling nonlinear systems or if the system possesses nonlinearities to any degree.

Generally, robots are machines that behave and carry out tasks like a human being. Lately, the industry is moving from automation to robot system, to increase productivity, reduce waste and to deliver uniform quality. Robots are mainly deployed to hostile environment such as in atomic plant to handle radioactive material, construct and repair space station and satellites, nursing and aiding patient in medical field, heavy earth moving equipment and many more.

The fundamental reason for engaging a robot is to eliminate the human operator. This reason is not just centered on safety but to save labor and reduce cost. Other classes of applications of concern are situations where human interference has negative impact especially on the products such as in food handling, semiconductor handling, pharmaceuticals and so on. Also, there are situations whereby the product has negative impact on the human such as;

radioactive product. Apparently, robots are alternative choice to human operators where the dangers are repetitive strain syndrome, places where the machines used are quite dangerous, like; presses, winders. Also, working with materials which might be harmful in the short or long term

According to Psaltis, Sideris and Yumanura, (1998), some specific form of adaptive control is meant to be carried out by neural controllers. The controllers are in the form of multilayer perceptron having interconnections between the neurons as the strength of the adaptable parameters. In a nutshell, controllers designed using NN architecture has the following characteristics (Psaltis, Sideris and Yumanura, 1998):

1. Ability to effectively utilize a large number of sensory information,
2. Have the capabilities to process data collectively
3. Must be adaptive by default.

NN Architecture: Multilayer Perceptron

MLN: The most popular NN architecture is the multi-layer perceptron. The network is made up of more than one layer comprising the input layer which can accept many inputs at a time, the hidden layer and the output layer as shown in Figure 1.1. Both the output and the hidden layers are made up of number of nodes that connect the neuron. However, the input layer has connection to the outside world and establishes a direct link to the inputs of the hidden layer.

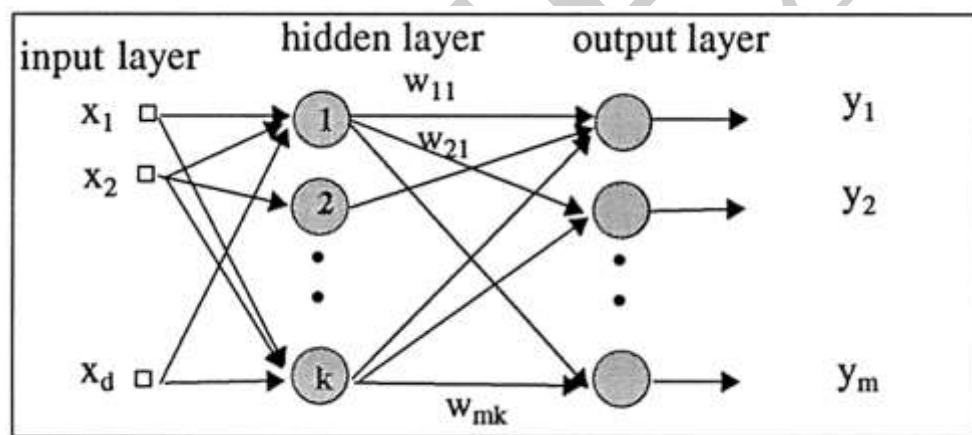


Figure 1.1: Multi-layer NN

The outputs of each node in a layer are connected to the inputs of all the nodes in the subsequent layer. Data flows through the network in one direction only, from input to output hence, this type of network is called a feed-forward network (Wallisch and Hatsopoulos, 2014). Log-sigmoid transfer function is most suitable for multilayer networks. The neural network transfer functions are shown in Figures 1.2 to 1.4 (Beale, Hagan, and Demuth, 2015).

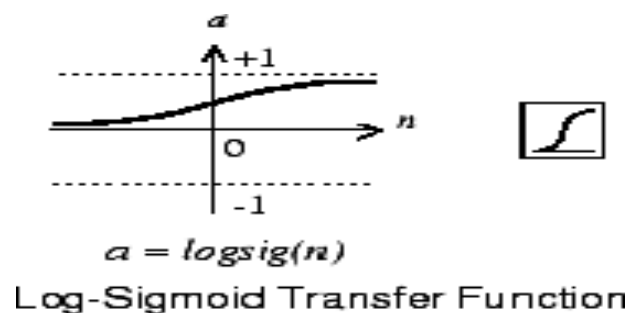
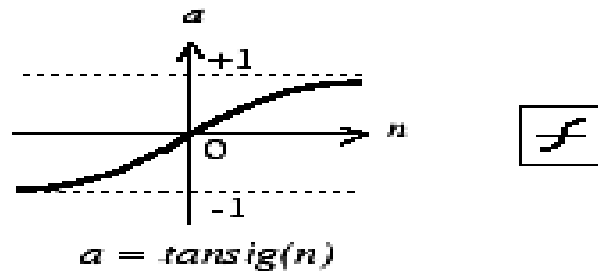


Figure 1.2: Log-Sigmoid Transfer Function (MathsWork.com, 2017)

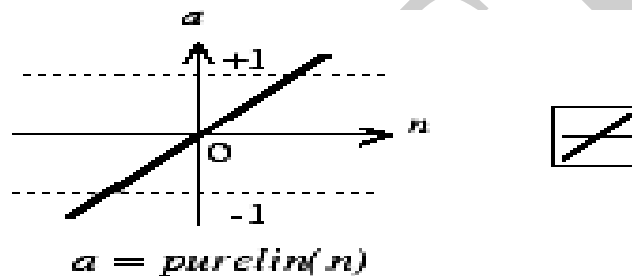
Multilayer perceptron can as well use the tan-sigmoid transfer function but that is occasional. Sigmoid output neurons are often used for pattern recognition problems, while linear output neurons are used for function fitting problems (Grace and Thompson, 2013).



Tan-Sigmoid Transfer Function

Figure 1.3: Tan-Sigmoid Transfer Function (MathsWork.com, 2017)

The linear transfer function (purelin) is shown in figure 2.17.



Linear Transfer Function

Figure 1.4: Linear Transfer Function (MathsWork.com, 2017)

The three transfer functions described here are the three most-commonly used, especially for multilayer networks. There are other differentiable transfer functions that can be created if desired. There are typically two steps involved when using NNs for control as reported by Beale, Hagan and Demuth, (2015):

1. System identification
2. Control design.

For the purpose of this work, discussion on NN control system will focus more on model reference adaptive control (MRAC).

2.0 Related Works Reviews

The following reviews are presented to further strengthen the background of this work for a deeper knowledge.

Jiang and Ishita (2008) did a work titled, A Neural Network Controller for Trajectory Control of Industrial Robot Manipulators. Based on neural network technology and linear feedback approach for tracking a planned trajectory, a new control scheme was proposed for industrial manipulators. With two parallel subsystems designed separately, the control system was designed such that one was a linear control and the other neural network control. The former was designed for trajectory tracking error regulation while the later for force/torque generation required by the designed dynamic trajectory. Based on simplified dynamic model of the robot, a learning law for online weight updating of the neural network controller was derived. A Direct Drive (DD) SCARA type industrial robot arm AdeptOne was used as an

application example for trajectory tracking control experiments. Simulation and experimental results were used to confirm the effectiveness and usefulness of the proposed control system. Pajaziti and Cana (2014) in Robotic Arm Control with Neural Networks Using Genetic Algorithm Optimization Approach, proposed the use of structural genetic algorithm in optimizing neural network to control the joint movements of robotic arm. The robotic arm was modeled in 3D and simulated in real-time in MATLAB. Neural Networks were found to provide a simple and effective to control the robot tasks. The significance of this method was verified by computer simulation results. A combination of Genetic Algorithm optimization method and Neural Networks for the given robotic arm with 5 D.O.F. showed that the base joint movements overshooting time without controller was about 0.5 seconds, while that with Neural Network controller (optimized with Genetic Algorithm) was about 0.2 seconds, and the population size of 150 gave best results. This shows the preference of NN in the control of robot arm movement.

Fateh et al, (2014) in Adaptive RBF Network Control for Robot Manipulators, proposed a controller which employs a simple Gaussian Radial-Basis-Function network (RBF network) as an uncertainty estimator. The proposed network included a hidden layer with one node, two inputs and a single output. The proposed estimator is simpler, less computational and more effective compared to other model-free estimators such as multilayer neural networks and fuzzy systems. Using an adaptation law derived by stability analysis weights of the RBF network could be tuned online. It is a voltage based control instead of a torque-based control. Simulation results showed the efficiency of the proposed control approach over robot manipulator driven by permanent magnet DC motors.

These reviews have shown the advantage and efficiency of using NN in the control of robot arm for improved operational and real-time industrial applications.

3.0 Methodology

The methodology applied in the actualization of the objective of this work is presented as described in this section.

3.1 Structure of the ANN Controller

, the design of ANN incorporated a Feedforward Neural Network (FFNN), which is made up of one input layer, a hidden layer and an output layer. The layers respectively consist of number of neurons. Each neuron has two functions:

- i. Summing up all the outputs from the previous layers, and then, multiplying it by the corresponding weights.
- ii. Performing the nonlinear sigmoidal or linear function on the sum

During training, errors are back-propagated and also minimized using least mean square algorithm. The basis for weights connection between the input and hidden layers are on the fact that errors in the output determine the measures of the hidden layer output errors. This adjustment of weights between the layers and recalculating the output in an iterative process is continued till the error falls below a tolerable level.

Mapping: here the input vectors are mapped to the output vectors by the Feedforward Back-propagation network. Before the training begins, pairs of input and desired output vectors that will be used in training the network are chosen. As soon as the training is completed and the weights set, at this time the system has learnt, then the network is used to determine or find outputs for new inputs. It could be noted that the dimension of the input vector determines the number of neurons in the input layer, and the dimension of the outputs determines the number of neurons in the output layer respectively.

Layout: This is based on the same fact mentioned above that the dimensions of the input and out patterns determine the number of neurons in the input and output layer. The network has three fields of neurons:

- i. One field for the input layer (Field A)
- ii. Another for the hidden processing element (Field B) and
- iii. The last for the output neurons (Field C)

The layers are connected and the connections are for feed-forward activity. They are connected in a manner that every neuron in A connects to B and those from B connect to C. also it should be noted that there are two sets of weights:

- i. The ones responsible for neuron activation in the hidden layer and
- ii. Thoseresponsible for neuron activation in the output layer.

To design the neural network controller, first the reference model of the plant (manipulator) to be controlled must be defined mathematically which we use the nonlinear auto regressive model (NARMA) equation in Martins et al (2013) to present;

$$y(k+d)=N(y(k),y(k-1),\dots,y(k-n+1),u(k),u(k-1),\dots,u(k-n+1)) \quad (3.1)$$

Where $u(k)$ is the system input, N is the nonlinear function, and $y(k)$ is the system output

$$\hat{y}(k+d)=f(y(k),y(k-1),\dots,y(k-n+1),u(k-1),\dots,u(k-m+1))+g(y(k),y(k-1),\dots,y(k-n+1),u(k-1),\dots,u(k-m+1))\cdot u(k) \quad (3.2)$$

To implement the controller the reference model (non-linear auto regressive model) and the approximate non-linear auto-regressive model (neural network plant model) were considered, hence, the neural network controller for the plant manipulator is as (Martins et al., 2013);

$$u(k+1) = \frac{y_r(k+d) - f[y(k), \dots, y(k-n+1), u(k), \dots, u(k-n+1)]}{g[y(k), \dots, y(k-n+1), u(k), \dots, u(k-n+1)]} \quad (3.3)$$

The structure of the NN controller and the NN plant were constructed using MathsWork NN toolbox in Simulink as shown in figures 3.1 and 3.3. The three inputs at three successive intervals were as seen as r , u and y . Time lag between the inputs is a unit delay. The tansig activation function was used at the middle layer, which calculates the middle layer's output from the net input from the input layer. It was used because it can handle values between -1 and +1. This is good in case there is a negative input value. At the output layer, the purelin function was used because it has the capability of handling many inputs and producing a single output

Now that the system has been defined, and the neural network will be trained to approximate the function (N) so as to generate a non-linear approximate model of the NN controller and plant (robot arm) (Martins et al., 2013). According to Okafor et al., (2017); the neural model reference adaptive control architecture used two neural networks: a controller network and a plant model network. The plant model is identified first, and then the controller is trained so that the plant output follows the reference model output. The reference model specifies the ideal response of the adaptive control system to the external command input or in other way provides the desired control system performance. The reference model will also help the NN controller to avoid having the trajectory to be tracked changed too rapidly. The NN control architecture is shown in Fig 3.3.

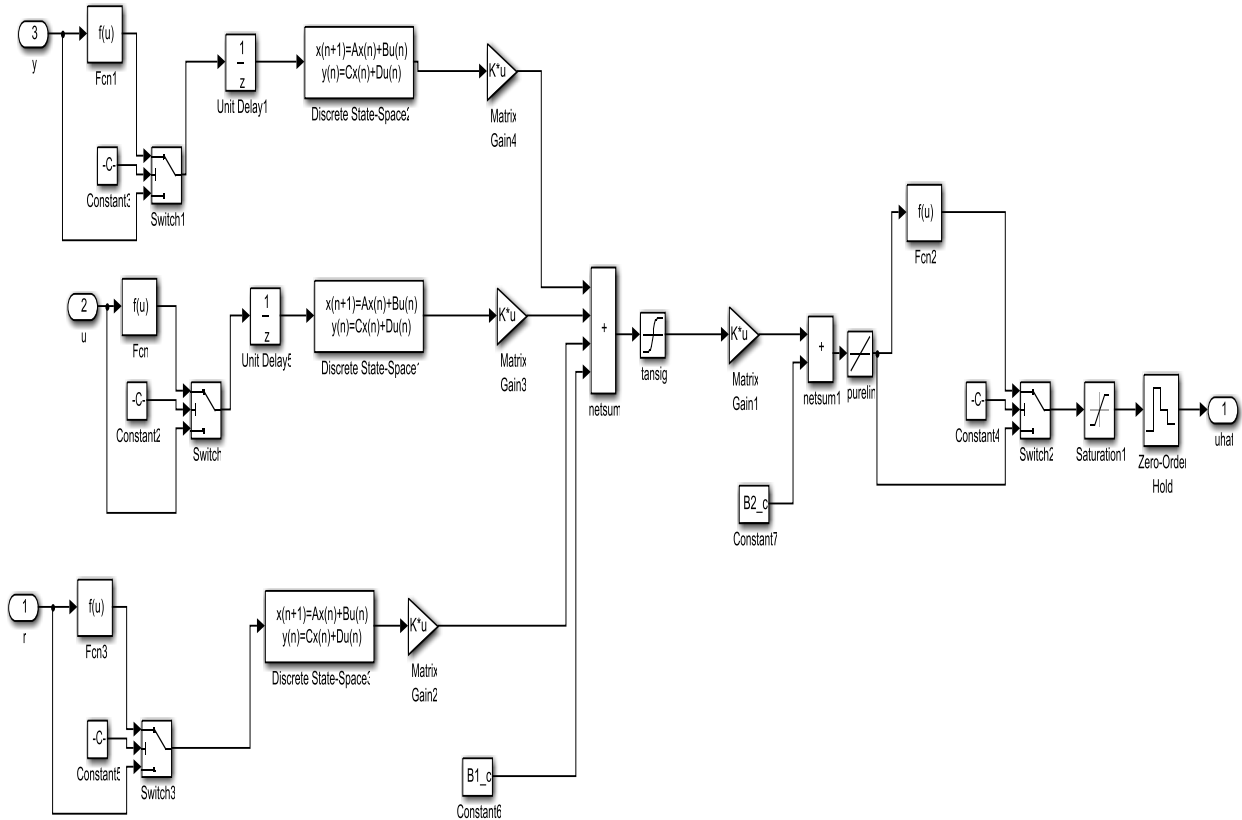


Figure 3.1: Result of the neural network architecture

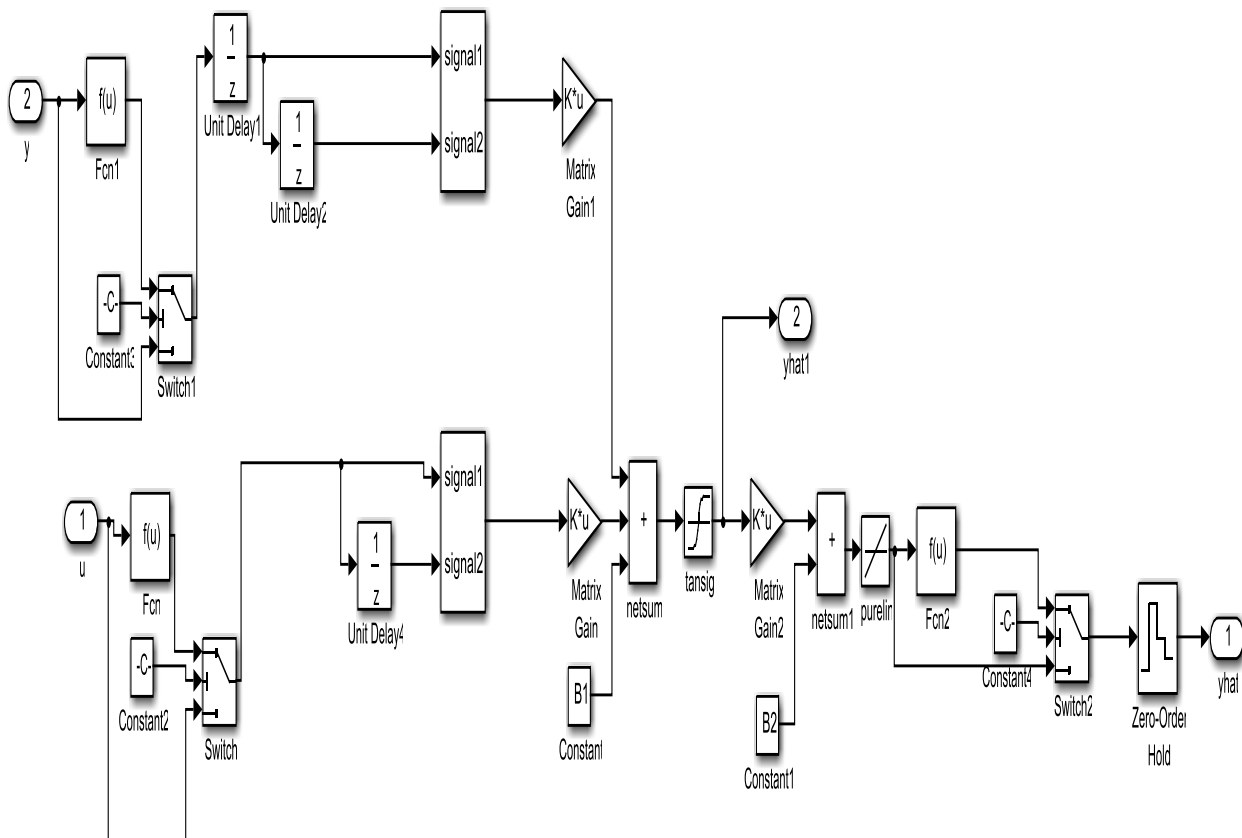


Figure 3.2: NN Model of the robot arm

From the figure, it could be observed that the output of the plant does not follow the reference input signal. The implication is that either of the two controllable parameters which is the speed or position of the robot arm was in error. It means that there is either no accuracy in the position or precision in the movement effectively, resulting to instability such as vibration the system. So to avoid such situation, the NN model of the plant needs to be trained, in order to learn from a predetermined desired input data and out data.

Now that the controller is designed, it will be used to control the manipulator dynamics. The NN was used in carrying out simulation using sinewave as reference input. The essence of the simulation is to see if the NN controller is capable of handling nonlinear effects of the system as a result of over voltage or current. This vibration is due to the variation in the amplitude of the applied signal that translates into mechanical vibration generated as a result of the vibration on the links. The result of the simulation is presented in Figure 4.2. Here, the neural network controller is shown in green patterns tracking the reference model of the plants.

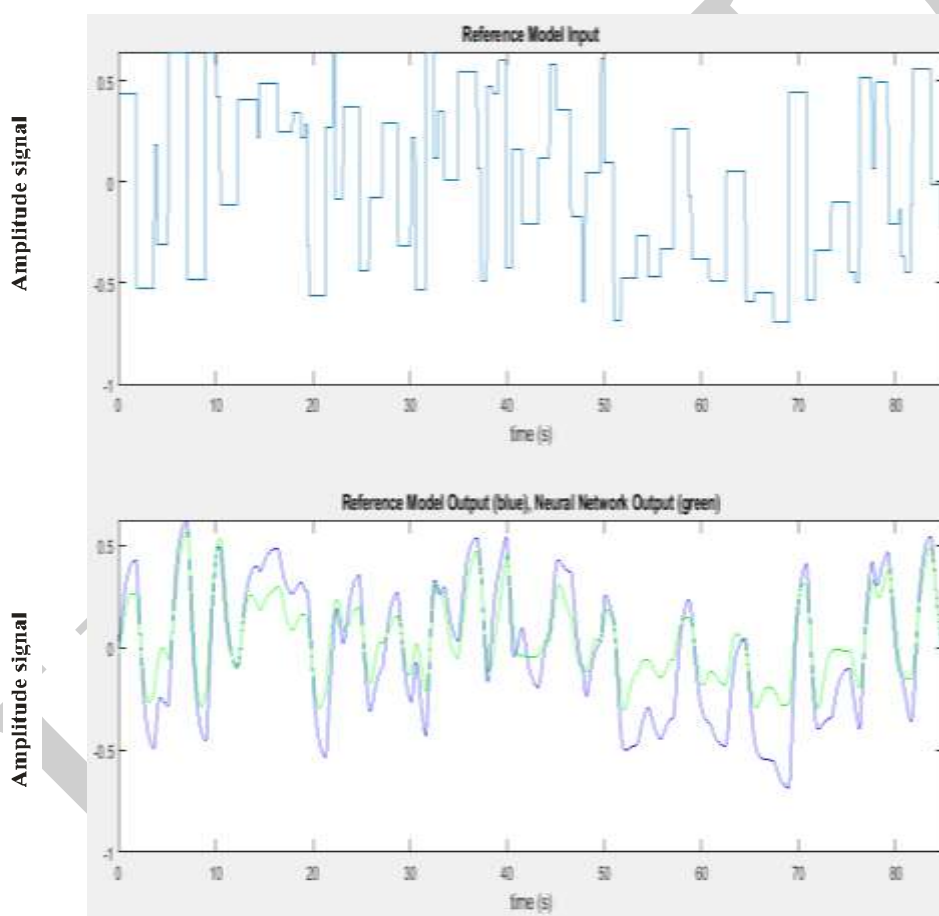


Figure 4.2: Neural network controller reference model output

From Figure 4.2, it could be observed that the NN output is trying to follow the reference signal closely but, because a better algorithm was not used in the training, the desired signal (output) continued to deviate from the reference signal input. To solve problem, the backpropagation algorithm was used in training the network.

Training of the Network: Mathematical Approach

It was said that the Feedforward Back-propagation undergoes supervised learning with a finite number of patterns consisting of an input pattern and a desired output pattern (Valluru, 1995). At the input layer, the input pattern is presented. Then, the input layer neurons pass the

activations to the next layer neurons i.e. those in the hidden layer. The outputs of the hidden layer neurons are obtained by introducing a bias, and also a threshold function, with activations determined by the weights and the inputs. The output from the hidden layer becomes the input to the output neurons, which process the input using an optional bias and a threshold function. Then the final output of the network is determined by the activations from the output layer.

The input and output of this network is guided by some basic equations. The overall input of the j^{th} neuron at the time instant n for the hidden layer is as given by (Haykins, 1999):

$$S_j^h(n) = \sum_{i=1}^N W_{ij}^h(n) I_i(n). \quad (4.1)$$

where W_{ij}^h is the connection from node to node (weight) between the i^{th} neuron at the input layer and the j^{th} neuron at the hidden layer. The I_i is the i^{th} input, and N is the number of inputs. Then, the output from the j^{th} neuron from the hidden layer at n^{th} instant is as given by (Haykins, 1999):

$$O_j^h(n) = f^h[S_j^h(n) + B_j^h(n)] \quad (4.2)$$

From equation (3.61), B_j^h is the bias of the j^{th} neuron and f^h is the activation function acting on each neuron at the hidden layer. The activation function can be tan sigmoidal, log sigmoidal or linear. The functions are described as follow:

$$\text{tansig}(x) = \frac{1 - e^{-2x}}{1 + e^{-x}} \quad (4.3)$$

$$\text{logsig}(x) = \frac{1}{1 + e^{-x}} \quad (4.4)$$

$$\text{linear}(x) = x \quad (\text{Haykins, 1999}) \quad (4.5)$$

In the above equations, x represents the input to the activation function. It follows that the net input of the k^{th} neuron of the output layer at time instant n is given by:

$$S_k^o(n) = \sum_{j=1}^M w_{jk}^o(n) O_j^h(n) \quad (\text{Haykins, 1999}) \quad (4.6)$$

where M is the number of neurons in the hidden layer and $w_{jk}^o(n)$ is the weight between the j^{th} neuron at the hidden layer and k^{th} neuron at the output layer respectively. It therefore followed also that output from the k^{th} neuron at the output layer at time instant n can be presented in the form:

$$O_k^o(n) = f^o[S_k^o(n) + B_k^o(n)] \quad (\text{Haykins, 1999}) \quad (4.7)$$

Where f^o = the activation function of the output layer and

$B_k^o(n)$ = the bias of the k^{th} neuron at the output layer.

It is also important to put into consideration how the weight is updated at various levels during the network training. To achieve that, there is a basic equation that describes the updating of the weight through the error signal at the output of the neuron k at the iteration and it is given by:

$$e_k(n) = d_k(n) - O_k^o(n) \quad (4.8)$$

Where $d_k(n)$ represents the desired output for neuron k . Figure 3.19 shows the flowchart of the training process of the system in artificial neural network.

Figure 4.3 shows the flow chart for the ANN training.

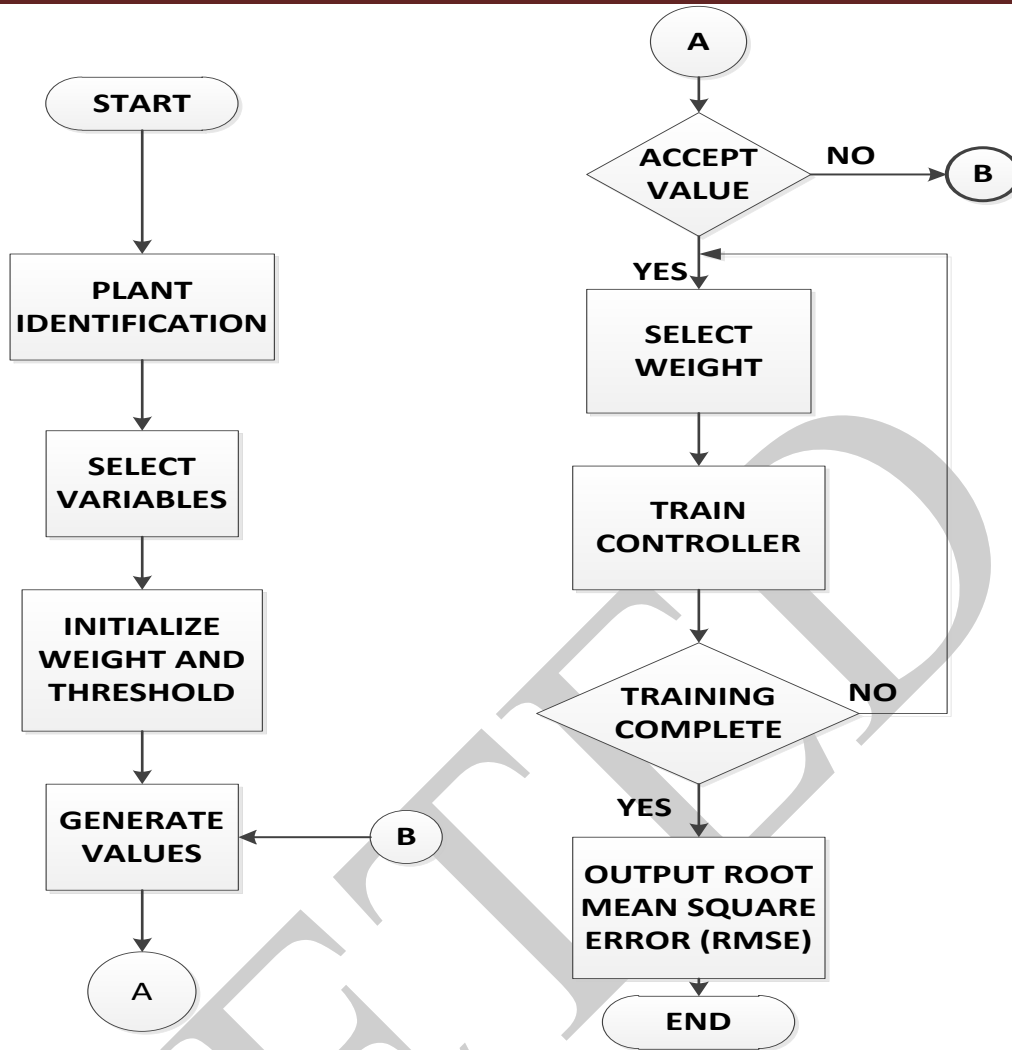


Figure 4.3: Flowchart for ANN Training

In training the network, the aim is to achieve optimal number of hidden layer neurons and also the learning parameter (Okafor, 2017). So, through training of different combination of hidden layer neurons and the learning parameter, the optimal number of hidden layer neurons and the learning parameter were obtained. The following parameters as shown in table 4.1 were used.

Table 4.1: Neural Network Parameters

Controller Training epochs	10
Size of hidden layers	10
Controller training segments	30
No. delayed reference input	2
Maximum plant output	3.1
Maximum plant input	15
Number of non-hidden layers	2
Maximum interval per sec	2
No. delayed controller output	1
No. delayed plant output	2
Minimum reference value	-0.7
Maximum reference value	0.7

Table 4.2: Manipulator joint configuration

Number of joints	Number of dc motor per joint	Angles (degree)
Angular position (t)	1	60
bicep (b)	2	10
Forearm (f)	1	60
Wrist (w)	1	90
Hand (h)	1	90
Gripper (g)	1	60

4.1 Neural Network Training: MATLAB Simulation Results and Analyses

The offline training of the controller was performed with this training data using Back-propagation algorithm as proposed in chapter three. The back-propagation training algorithm is based on the principle of minimization of a cost function of the outputs and the target of the FFNN. The net input of the j^{th} neuron of the hidden layer at the time instant n is given in equation (4.6), while the output from the j^{th} neuron from the hidden layer at n^{th} instant is given in equation (4.7). Here, the aim is to achieve optimal number of hidden layer neurons and also the learning parameter. So, through training of different combination of hidden layer neurons and the learning parameter, the optimal number of hidden layer neurons and the learning parameter were obtained. The following parameters as shown in tables 4.1 and 4.2 were used, which are parameters of the plant and controller for the optimal set. The objective is to train the controller so that the plant (robot arm) follows the reference model. The sampling interval used was 0.05 seconds.

Training of the neural network controller and the plant is a straightforward process, which can be achieved using the following steps (Beale, Hagan, and Demuth, 2015):

1. Call up the already-designed model, which could be in .slx file format. The plant to be controlled must be identified before it can be controlled. So, it is a must that the neural network plant model be developed before the controller can be used.
2. Generate the training data by applying series of random step inputs speed pattern to the Simulink plant model (Beale, Hagan, and Demuth, 2015).
3. Once the generated data is accepted then, training of the plant begins. The training proceeds according to training algorithm. The difference between the plant output and reference model output establishes the error. The basic equation that describes the updating of the weight through the error signal at the output of the neuron k was given as

$$e(t) = SP - PV \quad (4.9)$$

Where SP: Set point; PV: Process variable; P_0 :offset value

4. Once the plant training is completed, next is to return to the control system window and start simulation i.e. training the controller. The training program introduces a segment of data to the network and trains the network for a predetermined number of iterations (scribd.com, 2017). This process continues, one segment at a time, until the entire training set has been presented to the network (MathsWork.com, 2017). Controller training can obviously be more time-consuming than plant model training (Siddique and Adeli, 2013). This is because the controller must be trained using *dynamic* back-propagation. The feedforward back-propagation algorithm flowchart describing the procedures involved in deploying the algorithm is shown in figure 4.4.

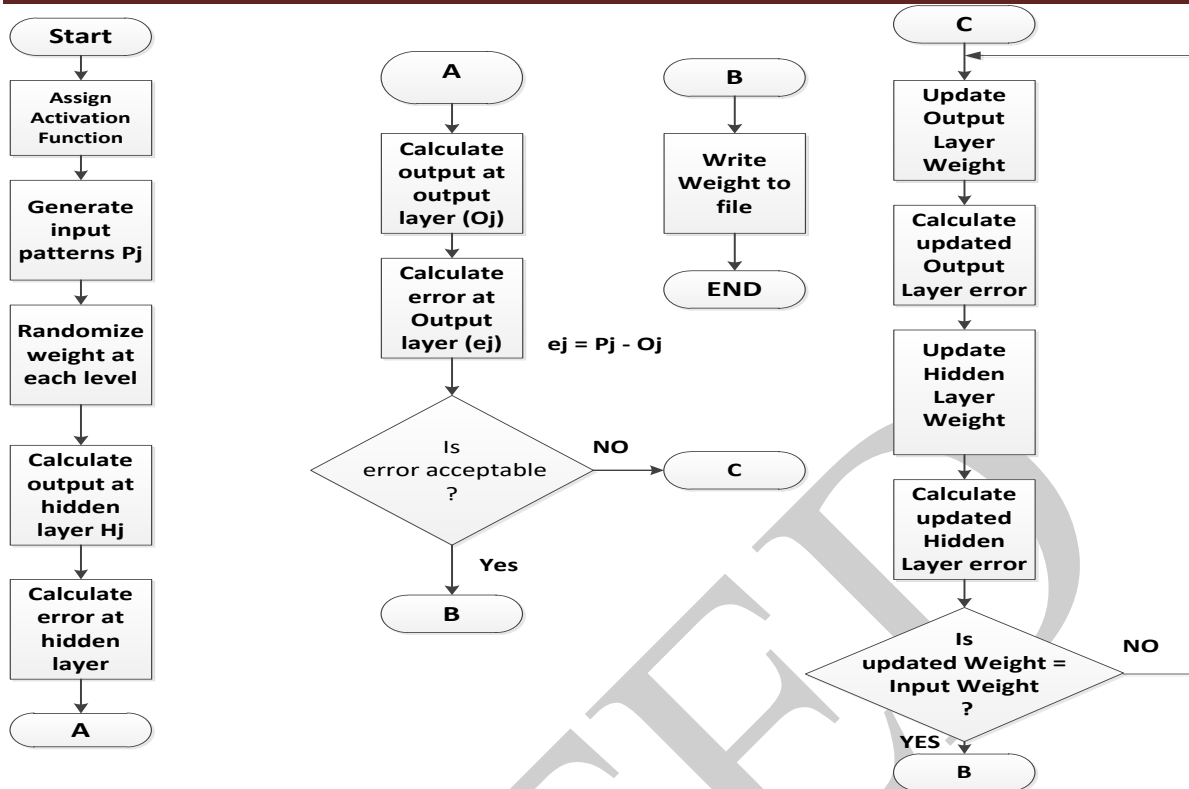


Fig 4.4: Flowchart for Feed forward Back-propagation Algorithm

From the NN training tool, it can be easy for the performance of the control system to be determined graphically as shown in figure 4.5. Once the robot arm motor is excited by an input signal, the output which is speed in terms of voltage is fed into the hybrid controller as an input. The terminal voltage is compared with the actual manipulator output for a common excitation signal. Then the mean square value of the error between the actual manipulator input and the estimated output voltage yields the performance error of the controller.

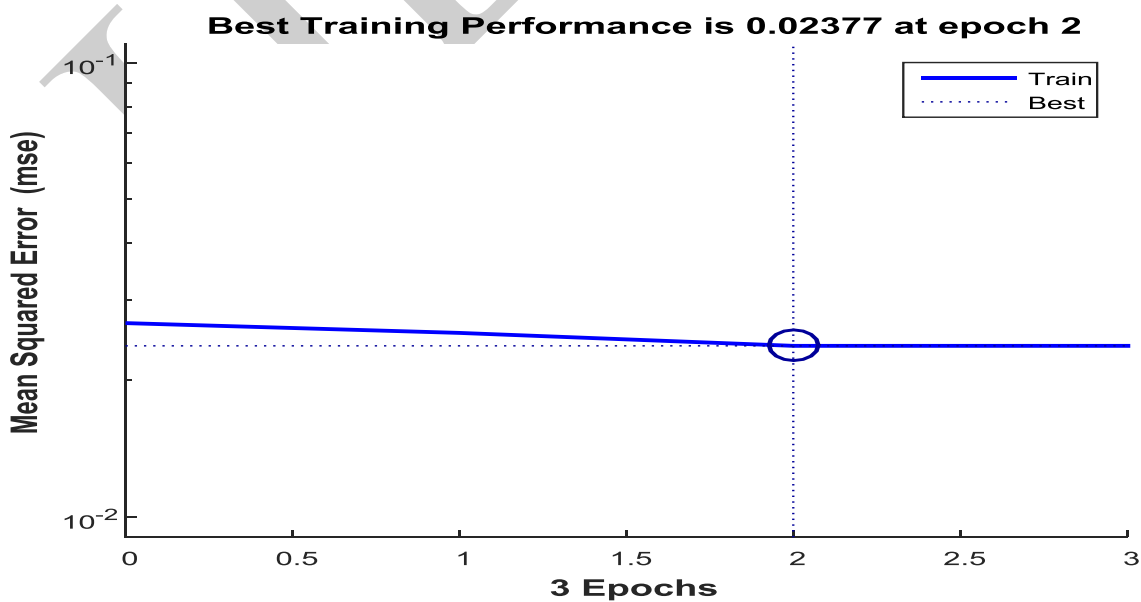


Fig 4.5: NN Training Performance

From figure 4.5, it can be seen that best training performance of 0.024 Root Mean Square Error (RSME) at epoch 2 was achieved. The beauty of this controller is that it can be trained over time to achieve the best result and each training state reports the mean error, gradient and the validation check. These parameters will guide the user to evaluate the accuracy of the controller at various training vectors (epoch). The error histogram chart is reported in figure 4.6.

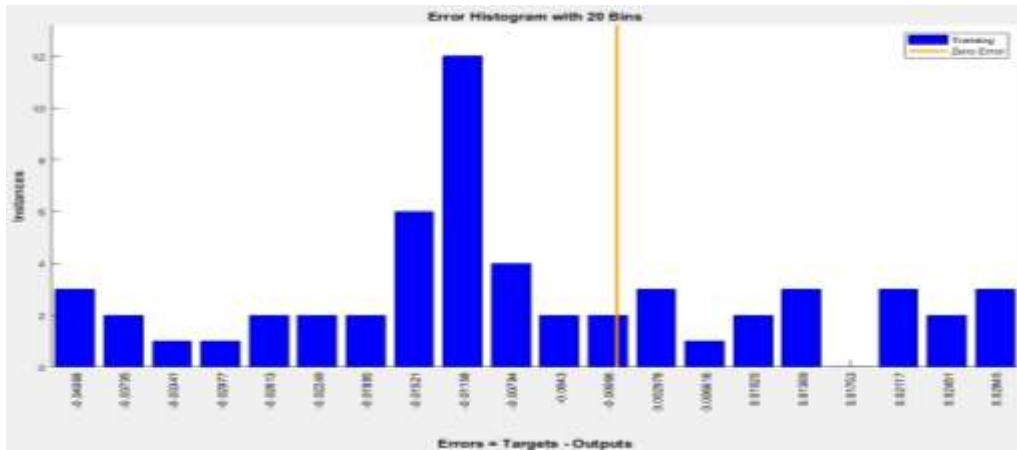


Figure 4.6: Neural network training error histogram

To accurately justify the result of the iteration, there is need for a regression analysis which will monitor the fittings of the neural network controller model (output), in line with the target reference model. This is achieved by creating a linear relationship between the output and the target. If the fitness is 100%, the linear relationship is $R=1$ which is the précised result for the manipulator. Although it is rare to achieve in practical, however if the relationship is $R=0$, then the performance of the neural network controller is very poor and need to be re-trained.

Figure 4.7 presents the regression plot for this work with the relationship between the target and output at $R=0.99862$ which means the fittings of the controller to the target is very accurate as required. This producing the resultant controlled effect on the manipulator as shown in figure 4.5. From the graph, the reference tracking result of the controller is in (blue) while that of the model to be controlled is in (red) patterns.

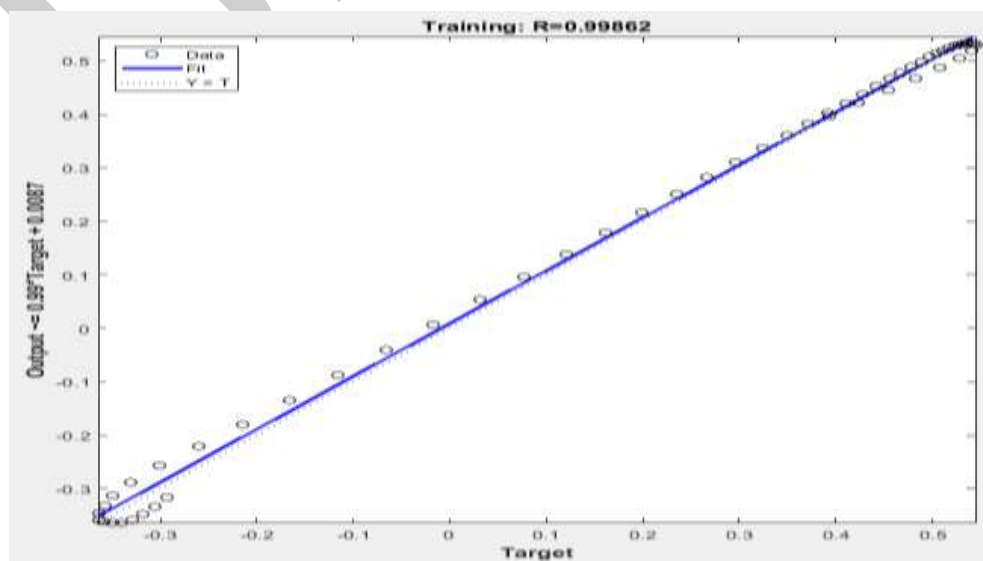


Figure 4.7: Neural network regression plot

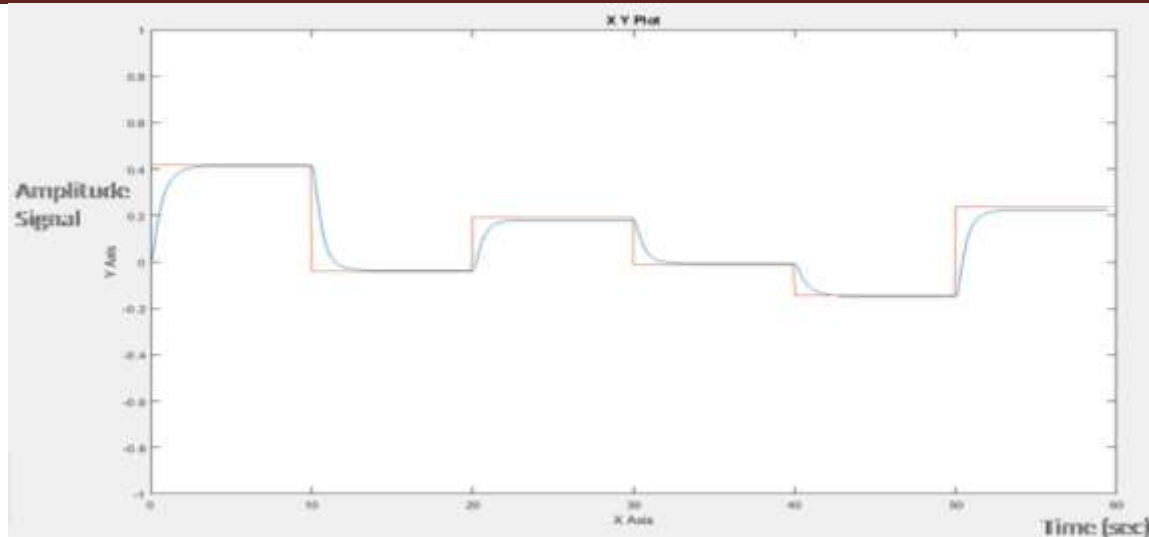


Figure 4.8: Neural network controller result

From the figure 4.8, it could be seen that the plant output followed the reference input which was the purpose of training the network.

5.0 Conclusion

The neural network training results to track and control the non-linear auto-regressive model (manipulator reference model) at various time series; the training performance result is presented in figure 4.5 with the measures of the number of times all the training vectors used for the network update. Best training performance of 0.024 Root Mean Square Error (RSME) reduction at epoch 2 was achieved. The implication of this result is that trained network i.e. the controller is capable to minimize the system error to almost zero. Figure 4.6 presents the neural network training error histogram with a very good data fitting which can be verified in the regression result in figure 4.4; showing the relationship (R) between the output and the target. The result as shown is dependable and reliable when compared to conventional robot arm control methods. The reliability is as a result of the training process applied to the NN controller, providing it with intelligence for a better response and stability for robot arm manipulator operations. Future work on neuro-fuzzy hybrid is envisaged and expected to yield a better control result as anticipated.

References

1. Zhao-Hui Jiang and Taiki Ishita (2008), A Neural Network Controller for Trajectory Control of Industrial Robot Manipulators, Journal of Computers, Vol. 3, No. 8, August 2008.
2. Anood Ibrahim, Reba Rachel Alexander, Mohammed Shahid Umar Sanghar and Royson.
3. Donate D'Souza, (2016), Control Systems in Robotics: A Review, International Journal of Engineering Inventions, Volume 5, Issue 5 [May 2016] PP: 29-38.
4. Pajaziti A. and H. Cana (2014), Robotic Arm Control with Neural Networks Using Genetic Algorithm Optimization Approach, International Journal of Mechanical and Mechatronics Engineering, Vol:8, No:8, 2014.
5. M. M. Fateh, S. M. Ahmadi and S. Khorashadizadeh, (2014), Adaptive RBF Network Control for Robot Manipulators, Journal of AI and Data Mining, Vol. 2, No. 2, 2014, 159-166.
6. Beale M. H., Hagan M. T, and Demuth H. B., (2015), "Neural Network Toolbox": The User's Guide. The MathWorks, Inc.

7. MathsWork.com., (2017), “Design Neural Network Predictive Controller in Simulink”: Retrieved from <https://www.mathworks.com/help/nnet/ug/design-neural-network-predictive-controller-in-simulink.html>.
8. Okafor P.U., Eneh I.I., Eneh P.C., (2017). “Improving The Control Of Dc Servomotor Speed And Position Using Model Reference Adaptive Control (MRAC)”. International Journal of Research in Engineering and Applied Sciences (IJREAS) Available online at <http://euroasiapub.org/journals.php> Vol. 7 Issue 8, ISSN (O): 2249-3905, ISSN(P): 2349-6525.
9. Okafor P.U., Eneh P.C., Arinze S.N., (2017). “Model Reference Adaptive Control (MRAC) Scheme For Eliminating Overshoot In Dc Servomotor”. International Journal of Advanced Research in IT and Engineering (ISSN: 2278-6244). Volume 6, Issue 3. Pp.14-30.
10. Siddique N., and Adeli H., (2013), “*Computational Intelligence*”; *Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing*: John Wiley & Sons.
11. Valluru B.R., (1995), “*C++ Neural Networks and Fuzzy Logic*”; MTBooks, IDG Books Worldwide, Inc. ISBN:n1558515526.