PERFORMANCE IMPROVEMENT IN INVERTER DESIGN BY USING PARALLEL DISTRIBUTED HYSTERISIS BAND PULSE WIDTH MODULATED CURRENT CONTROLLER IMPLEMENTED FOR A VECTOR CONTROLLED INDUCTION MOTOR

Sreedhar Madichetty[1], Prasanta Kumar Prusty[2]

[1]School of Electrical Engineering
KIIT University, Bhubaneswar, Orissa, INDIA-751024.
sreedhar.803@gmail.com[1],
prasanta.er@gmail.com[2]

ABSTRACT

The aim of the paper is to design a parallel distributed hysterisis band PWM current controller for a three level voltage source inverter (VSI) through an artificial neural network (ANN) approach. The ANN approach has been selected for this, since it has the potential to provide an improved method of deriving non-linear models which is complementary to conventional techniques. To illustrate the validity of this approach, an indirect vector controlled induction motor (IVCIM) drive has been considered as its application.For designing this, a feed forward neural network using the error back propagation training algorithm is selected. This network is trained so that it learns the nonlinear behavior of the conventional HBPWM controller. Thus the neural network acquires the features of an HBPWM controller and behaves as an PDHBPWM. The simulation model of an IVCIM drive employing PDHBPWM current controller is developed in SIMULINK/MATLAB environment. The performance of the proposed method is compared with conventional HBPWM current control scheme. From the observations made it is concluded that PDBPWM current controller is more efficient in terms of fault tolerance, switching loss and total harmonic distortion thus improving the performance of the drive.

Terms: VSI (Voltage Source Inverter), Vector Control,Hystersis Band Control,Parallel Distribution, Artificial Neural Network.

I. DESIGN OF PARALLEL DISTRIBUTED HBPWM CURRENT CONTROLLER

The controller of the inverter must have some nonlinear functions to convert the analogue inputs into binary outputs. For this reason, the on-off pattern controller of the inverter conventionally includes non linear functions such as hysterisis comparators. Among the various current control methods,The hysteresis control scheme provides excellent dynamic performance because it acts quickly.

The application of ANNs is recently growing in the areas of power electronics and drives. Artificial neural network concept is a synthetic approach [8] to actualize the non-linear data mapping of the hysteresis control scheme. A feed forward ANN basically implements nonlinear input-output mapping. Among several learning rules of ANN, the error back propagation algorithm [2] is well known and useful because it can be applied to multi layer networks. It learns the non linear data mapping from the given input and output data as teaching signals. The teaching signal should be determined carefully so as to include sufficient information to learn the essential characteristics of the desired data mapping.

Remarkable features of neural networks are both fast processing speed and fault tolerance to some overlook connections in the networksystem. Because of the parallel processing mechanism of neural networks, it is expected that the neural networks can execute the non-linear data mapping in short time and of the distributed network structure, the performance of the neural network may not be influenced by some miss connections in the network itself. A control system with multi input neural network may not be affected by partial fault in the system, because some other correct input data may compensate the influence of wrong input data.

II. FEEDFORWARD NEURAL NETWORK

The most commonly used feed forward; multilayer network is the back propagation-type network. The name “back propagation” comes from its training method. The structure of feed forward back propagation-type neural network is shown below in the Fig 1.

Error Back Propagation Training

Error Back propagation or Back Propagation is the most popular training method for a multi layer feed forward network. Basically it is a generalization of the delta learning rule developed for Adaline training. Input and output patterns are submitted sequentially during the back propagation training. If a pattern is submitted and its classification or association is determined to be erroneous the synaptic weights are adjusted so that the current least mean square classification error is reduced. The input-output mapping comparison of target and actual values, and adjustment, if needed, continue until all mapping examples from the training

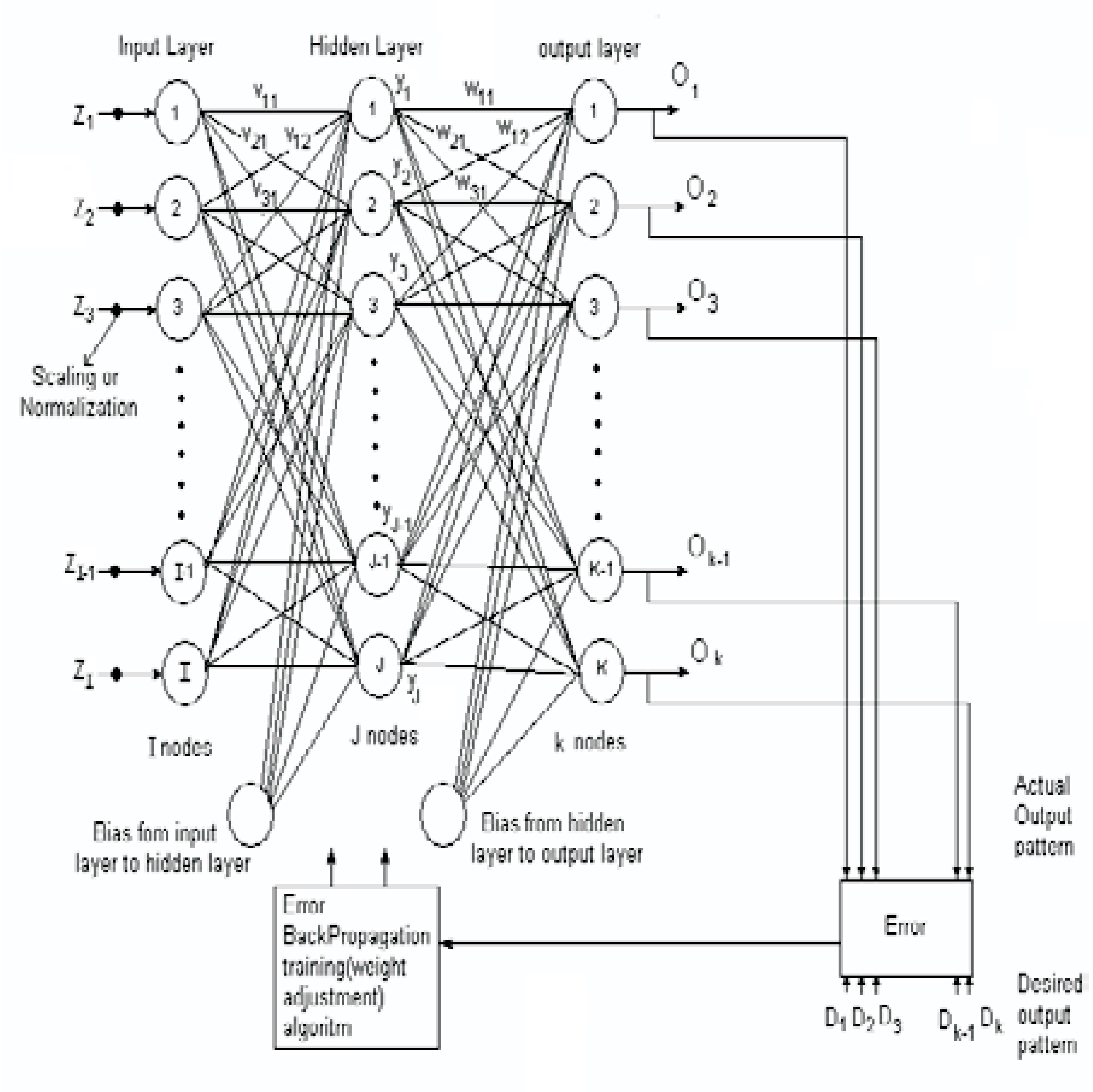


Fig. 1. Structure of Feed forward Neural Network showing Back Propagation training

set are learned within an acceptable overall error. Usually mapping error is cumulative and computed over the full training set. During the association or classification phase, the trained neural network itself operates in a feed forward manner. The weight adjustments enforced by the learning rules propagate exactly backward from the output layer through the hidden layers towards the input layer and hence the name back propagation. Fig 2 shows the general flowchart of the error back propagation training. In the beginning the network topology with the number of layers, number of neurons in the hidden layer(s), and activation function is selected. The input/output training data patterns can be gathered from the experiment or from the simulated system if the system model is available. ANN topology and initial weights play a very important role in training the network. The number of hidden layers and number of hidden layer neurons are selected by experience so that the network is trained satisfactorily within an acceptable overall error. Once a network is trained the synaptic weights remain constant and further training of the network does not alter the weights.

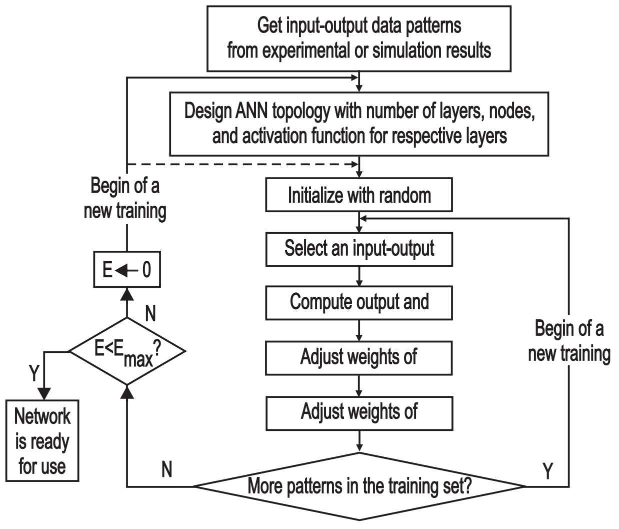

Get input-output data patterns from experimental or simulation results
Design ANN topology with number of layers, nodes, and activation function for respective layers
Begin of a new training
Initialize with random
Select an input-output
Compute output and
Adjust weights of
Adjust weights of
More patterns in the training set?
E ← 0
N
E<E_max?
Y
Network is ready for use
N
Y
Begin of a new training

Fig. 2. Flowchart for Error Back Propagation training of a Neural Network

III. CREATION AND TRAINING OF THE PDHBPWM CONTROLLER IN SIMULINK

The neural network chosen to emulate the HBPWM controller is a feed forward multilayer network. The network topology depends on the hysteresis band width selected. The simulation results presented in this paper are obtained by considering a band width of 0.02. The network has a 3-15-3 structure i.e. 3 input neurons, 15 hidden layer neurons and 3 output neurons. If a hysteresis band width of 0.5 is selected, the network will have a 3-3-3 structure. The hidden layer neurons are selected by trial and error so that the network is satisfactorily trained for the considered training data. Training samples [9] used to train the network should be sufficient so that the network learns the behavior of the hysterisis PWM controller. We have considered 460 training samples for this problem. The training samples are obtained by running the IVCIM drive using conventional HBPWM current controller. The error back propagation algorithm is used to update the weights so as to decrease the current errors.

The PDHBPWM inverter control method is shown in the Fig 3. The inputs to the PDHBPWM controller are three phase current errors and the outputs are the switching patterns to the PWM inverter as in the case of conventional HBPWM controller. k in the figure represents the normalization factor and is used for the purpose of scaling the current error input to the PDHBPWM controller.

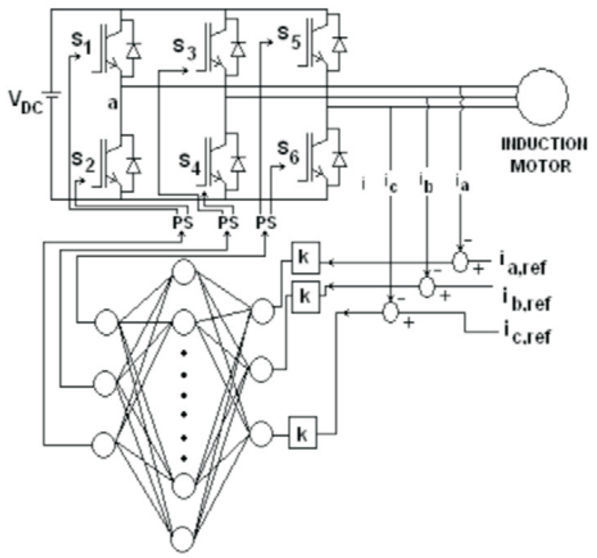


Fig. 3. PDHBPWM Inverter Control method

The training rule [7] is the same as the principle of operation of HBPWM current controller.

$| i_{m,\,ref} - i_m | < \varepsilon$ keeps the output pulse at the same state

$i_{m,\,ref} - i_m > \varepsilon$ let output pulse = 1 (high)

$i_{m,\,ref} - i_m < -\varepsilon$ let output pulse = 0 (low)

where $m = a, b, c$ phases and ε is the hysterisis band

The SIMULINK neural network toolbox consists of subsystems such as control systems, net input functions, transfer functions, weight functions. Each of these subsystems consists of functional blocks which are useful in developing SIMULINK models for different types of neural networks. MATLAB also supports Graphical User Interface (GUI) [3]. It is designed to be simple and user friendly. This window has its own work area, separate from the more familiar command line workspace. Thus, when using the GUI, we can "export" the GUI results to the (command line) workspace and also "import" results from the command line workspace to the GUI. This interface allows us to:

- Create different types of neural networks
- Enter data into the GUI
- Initialize, train, and simulate networks
- Export the training results from the GUI to the command line workspace
- Import data from the command line workspace to the GUI

Algorithm for Design of PDHBPWM Current Controller

STEP 1: Generation of training data for the neural network

STEP 2: Using GUI, design a feedforward neural network and initialize its weights with random values.

STEP 3: Select the error back propagation training algorithm for the training of the network

STEP 4: Train the feedforward network using the generated training data

STEP 5: If the MSE (Mean Squared Error) is below the acceptable value, Then training is complete and the network is ready for use Else, The network is not trained, go to Step 2.

Generation of Training Data to Train the Neural Network

Training in this case is required to learn something about the plant behavior i.e. the neural network must learn the non-linear behavior of the hysterisis current control.

The training data is generated by simulating the IVCIM drive using conventional HBPWM current controller. Connect ***To workspace*** blocks available in Sinks in SIMULINK library at the input and output of HBPWM current controller to store the generated training data in the workspace.

Procedure to Create and Train the PDHBPWM controller

Creation and training of PDHBPWM current controller using neural net toolbox can be implemented in the following steps:

STEP 1: The first step is to open the Network/Data Manager window. To open the Network/Data Manager window type the function ***nntool* in the command window in the main Matlab window and press enter. The Network/Data Manager window is shown below in the Fig 4.**

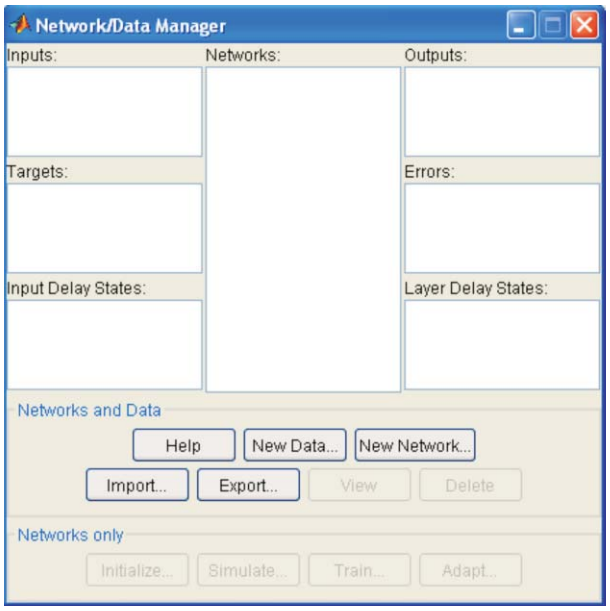
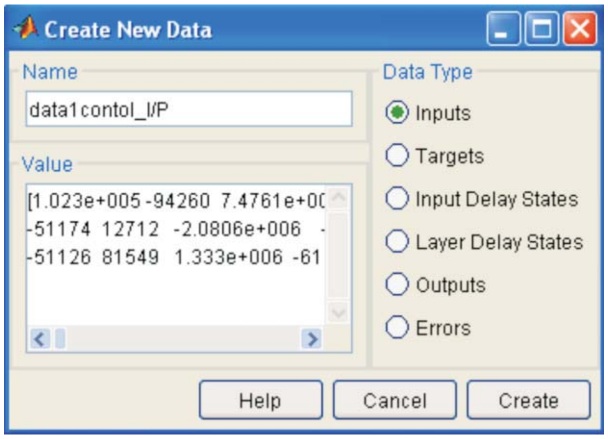


Fig. 4. Network/Data Manager window

STEP 2: Click on Help to get started on a new problem and to see descriptions of the buttons and lists.

Creation of input training data file *control_I/P:*

Define the network input file, say ***control_I/P***. Retrieve the training data from the workspace and consider 460 samples of input data. As the inputs to the HBPWM controller are three phase current errors and 460 samples of input have been considered, the input training matrix will be of the order 3×460. Thus, the network has a three-element input and 460 sets of such three-element vectors are presented to it in training. To define this data, click on New Data, and a new window, Create New Data appears. Set the Name to ***control_I/P***, Value to input training data, and make sure that Data Type is set to Inputs. The Create New Data window will then look like in the Fig 5.



Name: data1contol_I/P

Value:

```
[1.023e+005 -94260 7.4761e+00
-51174 12712 -2.0806e+006 -
-51126 81549 1.333e+006 -61
```

Data Type: [✓ Inputs] Targets Input Delay States Layer Delay States Outputs Errors

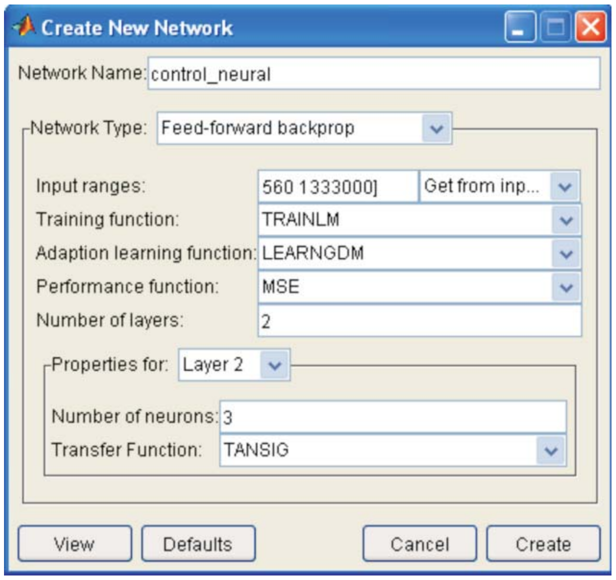
Fig. 5. Create New Data window

Now click Create to actually create the input file ***control_I/P***. The Network/Data Manager window comes up and ***control_ I/P*** shows as an input.

Creation of target/output training data file ***control_targ:***

Define the network output file, say ***control_targ.*** Retrieve the training data from the workspace and consider 460 samples of output data. The output matrix will be of the order 3×460. Click on New Data again, and this time enter the variable name ***control_targ***, specify the output training data under Value and click on Target under data type. Again click on Create and the resulting Network/Data Manager window that will now appear will have control_targ as a target as well as the previous control_ I/P as an input.

STEP 3: Now create a new network, called ***control_neural***. To do this, click on New Network, and a Create New Network window appears. Enter ***control_neural*** under Network Name. One can select the type of neural network, training and learning functions, performance function, number of layers for the network, number of neurons for each layer and transfer function in this window. Set the Network Type to Feed Forward Back Propagation, as that is the kind of network to be created. The input ranges can be set by entering numbers in that field, but it is easier to get them from the particular input data that is to be used. To do this, click on the down arrow at the right side of Input Range and click on control_I/P. Select the training and learning functions [3] as TRAINLM and LEARNGDM.TRAINLM is a network training function that updates weight and bias values according to Levenberg-Marquardt optimization. LEARNGDM function is gradient descent with momentum weight and bias learning function. Select Number of 1st and 2nd layer neurons as 15 and 3 respectively. Select TANSIG transfer function for both the layers. Now Create New Network window should look like in the Fig 6.



Network Name: control_neural

Network Type: Feed-forward backprop

Input ranges: 560 1333000]

Training function: TRAINLM

Adaption learning function: LEARNGDM

Performance function: MSE

Number of layers: 2

Properties for: Layer 2

Number of neurons: 3

Transfer Function: TANSIG

Fig. 6. Create New Network window

A view of the network can be obtained by clicking on View button. This is shown in the Fig 7.

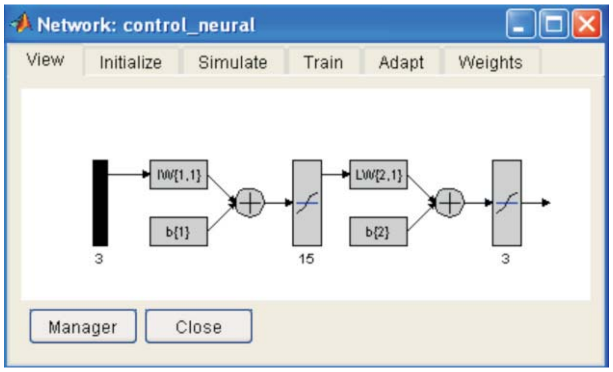
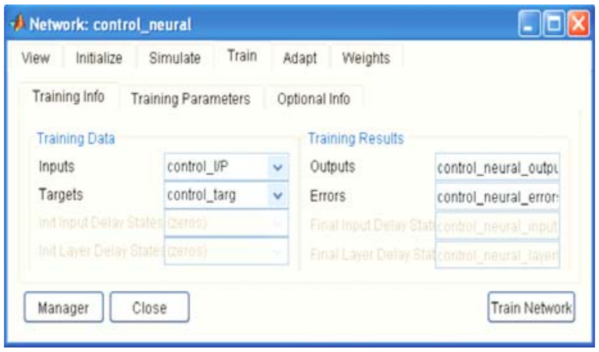


Fig. 7. View of the created network

Now click Create to generate the network. You will get back the Network/Data Manager window. Note that control_neural is now listed as a network.

Step 4: To train the network, click on control_neural to highlight it. Then click on Train. This leads to a new window labeled Network: control_neural. Specify the inputs and output by clicking on the left tab Training Info and selecting control_I/P from the pop-down list of inputs and control_targ from the pull-down list of targets. The Network: control_ neural window should look like the Fig 8.


Network: control_neural

Training Data		Training Results	
Inputs	control_I/P	Outputs	control_neural_outpu
Targets	control_targ	Errors	control_neural_error
Init Input Delay States	(zeros)	Final Input Delay State	control_neural_input
Init Layer Delay States	(zeros)	Final Layer Delay Stat	control_neural_layer

Fig. 8. Network: control_neural window

The Training Results are Outputs and Errors. They have the name control_neural appended to them. This makes them easy to identify later when they are exported to the command line. Now click on the Training Parameters tab. It shows parameters such as the epochs and error goal. These parameters can be changed at this point if required. Now click Train Network to train the backpropagation network. Then the window shown in the Fig 9 opens. For ideal training the performance must be equal to goal (usually equal to zero). In the present case the performance is reduced to 3.21722e-014 in 12,500 epochs which is nearly zero. The network will work satisfactorily as a hysterisis comparator only if the performance is reduced to such a small value. Output errors decrease in proportion to the performance index.

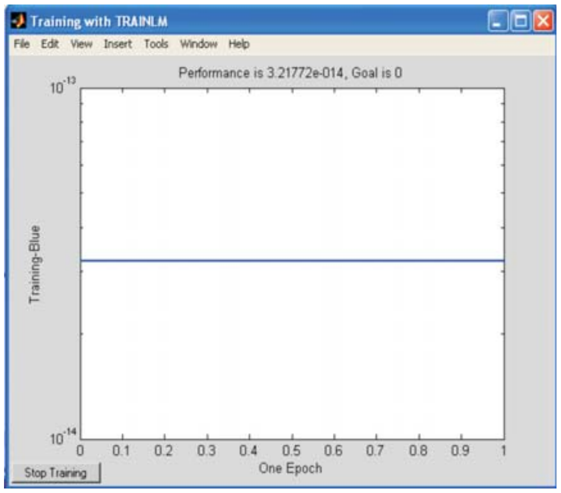

Performance is 3.21772e-014, Goal is 0

Fig. 9. Training window

IV. SIMULINK MODEL OF PDHBPWM CURRENT CONTROLLER

The SIMULINK block diagram of NHBPWM current controller is shown below in the Fig 10.

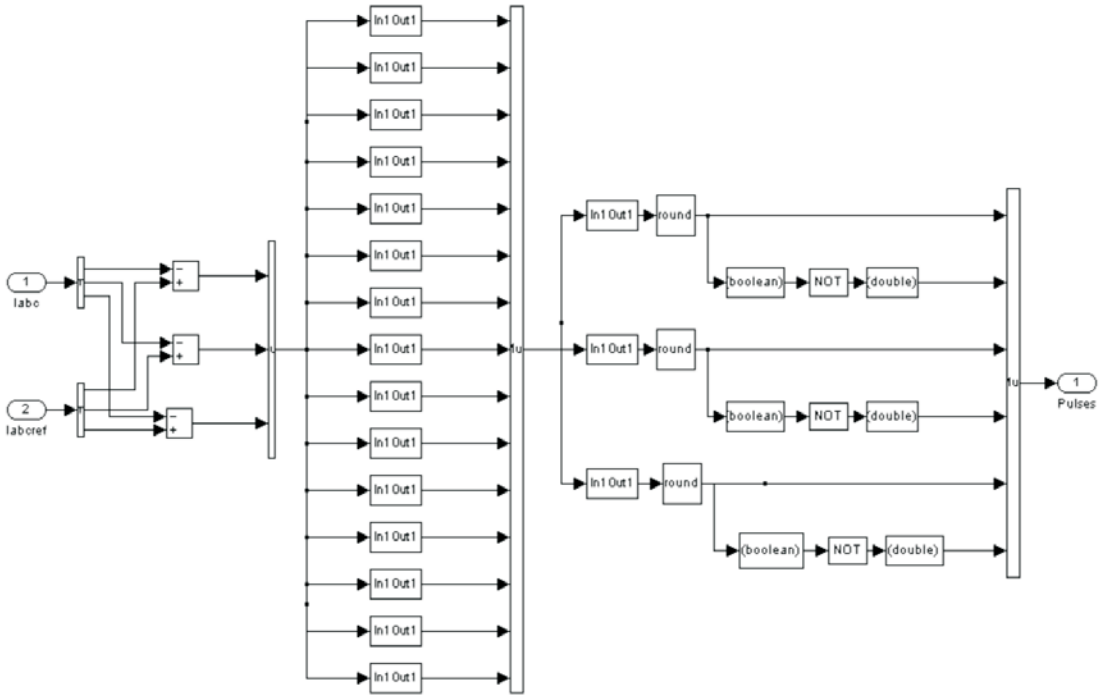


Fig. 10. Simulink block diagram of NHBPWM

The input layer consists of three phase current errors as input. The hidden layer consists of 15 neurons where each neuron is represented as a subsystem. Similarly each of the output neuron is represented by a subsystem. The subsystem model representing each neuron is shown in the Fig 11.

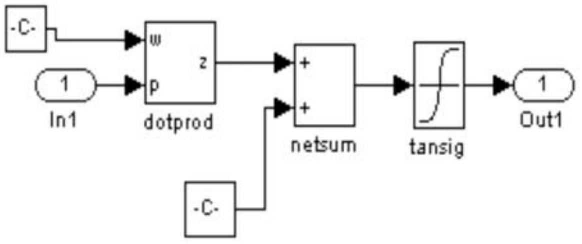


Fig. 11. Subsystem model of neuron

The neuron model in the Fig 11 is constructed according to the basic neuron model . Once a neural network is trained the weights of the network remains constant. Hence, the weights of the neural network after training are represented by the constant blocks in the Fig 11. The transfer function of each neuron unit is tan sigmoidal function.

V. SIMULINK MODEL OF IVCIM DRIVE USING PDHBPWM CONTROL

To perform a comparative evaluation of conventional and neural hysterisis current controller, the simulation model of an IVCIM drive using PDHBPWM current controller is also developed and simulated. The model is shown in the Fig 12. The block diagram is same as that of drive using conventional HBPWM current controller except that the conventional controller is replaced by PDHBPWM controller.

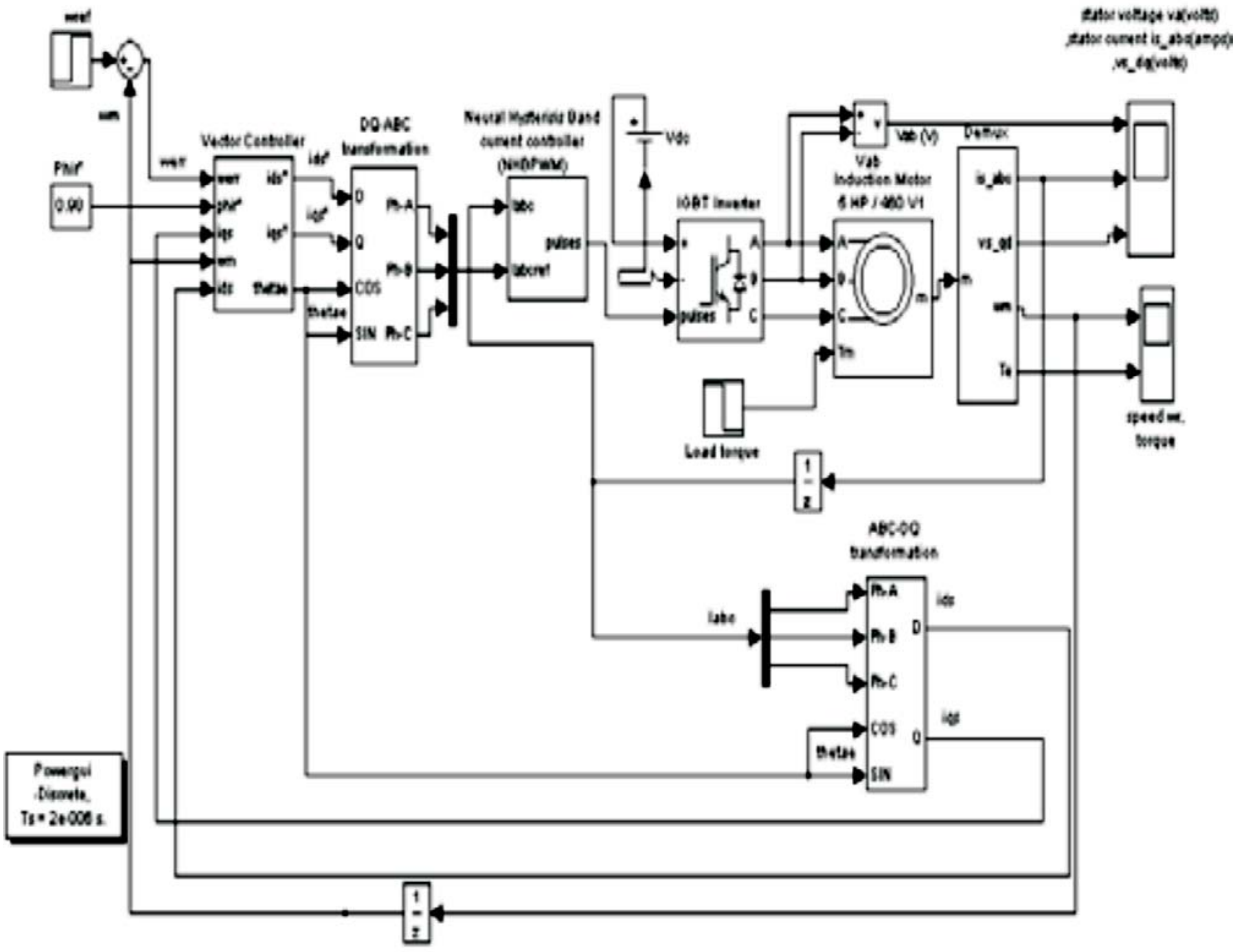


Fig. 12. SIMULINK Model of IVCIM drive using PDHBPWM Control

The reference inputs to the system are the reference speed (ω_{ref}) in radians per second, the flux linkages (phir*) and the load torque (Load torque). The required outputs are the developed electromagnetic torque (T_e) and the rotor speed (ω_m).

VI. SIMULATION RESULTS AND DISCUSSIONS

In order to illustrate the proposed PDHBPWM controller scheme an IM with the ratings specified. Simulation results of IVCIM drive using PDHBPWM controller are presented in this section.

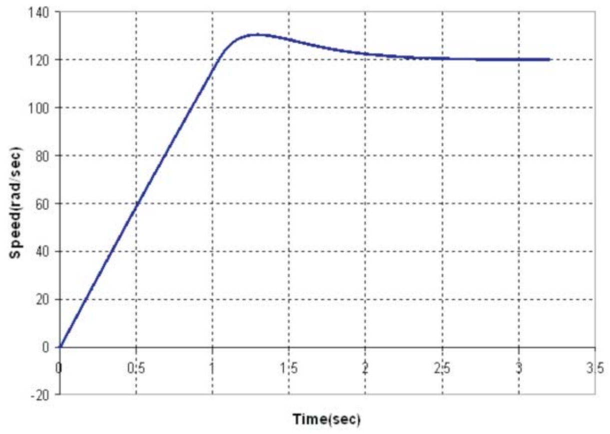


Fig. 13. Speed response of IVCIM using PDHBPWM controller without any disturbance

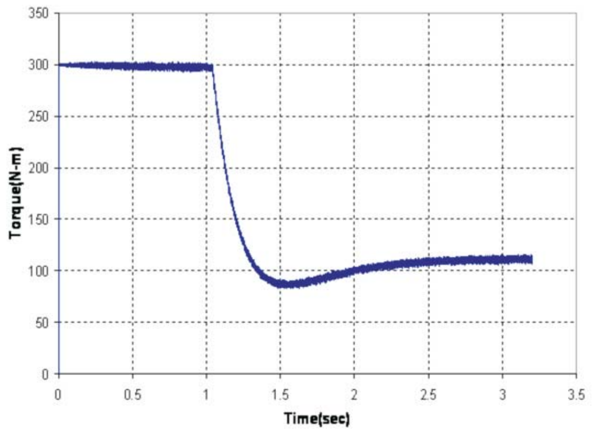


Fig. 14. Torque response of IVCIM using PDHBPWM controller without any disturbance

From the speed and left torque responses of the IVCIM shown in Figs 13, 14, 15 and 16 it is concluded that the performance of both the HBPWM and PDHBPWM controllers is the same.

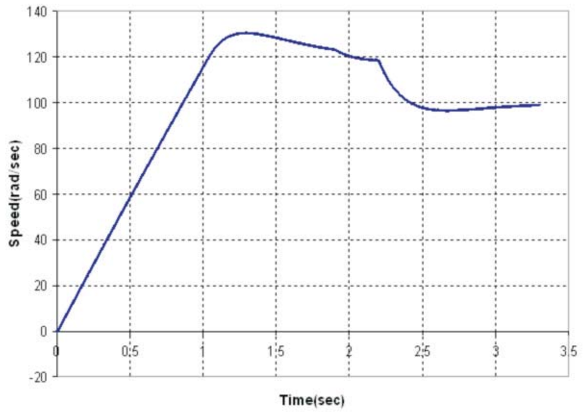


Fig. 15. Speed response of IVCIM using PDHBPWM controller with step disturbance

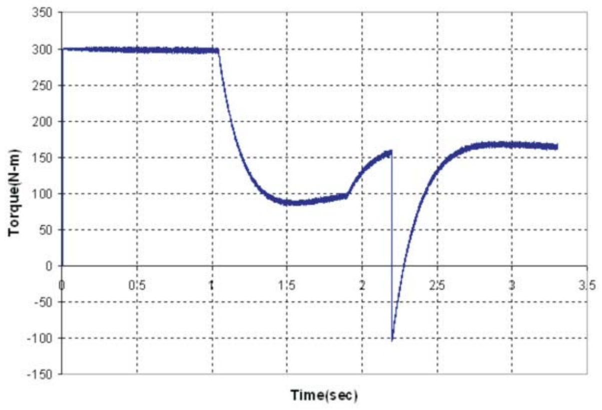


Fig. 16. Torque response of IVCIM using PDHBPWM controller with step disturbance

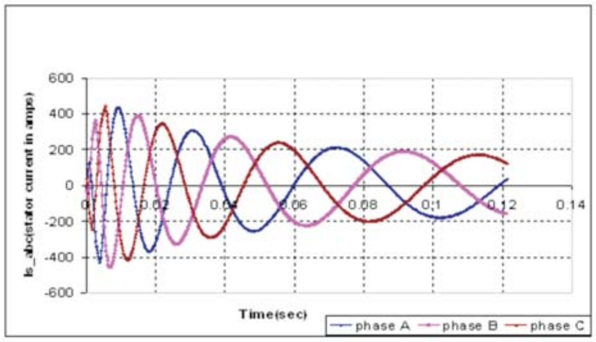


Fig. 17. (a) Stator current of IVCIM using PDHBPWM controller with k = 1

From the Figs 17(a) and 17(b) it is observed that the stator currents of the motor using PDHBPWM and HBPWM controllers respectively, with normalization factor k = 1, are smooth and identical.

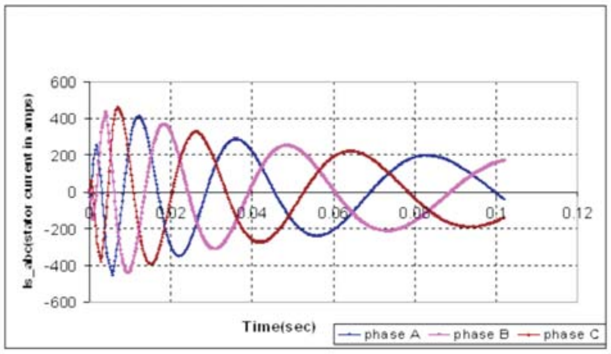


Fig. 17. (b) Stator current of IVCIM using HBPWM controller with k = 1

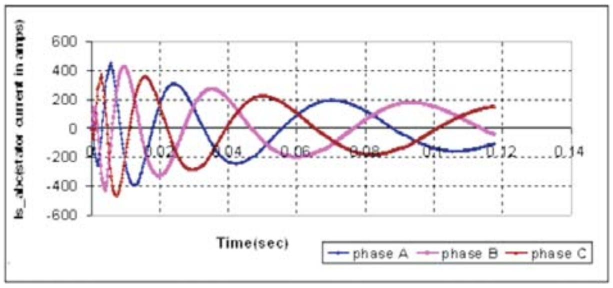


Fig. 18. (a) Stator current of IVCIM using PDHBPWM controller with k = 0.003

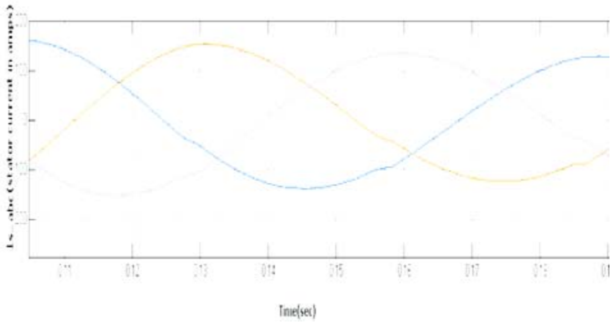


Fig. 18. (b) Zoomed portion of Stator current of IVCIM using PDHBPWM controller with k = 0.003

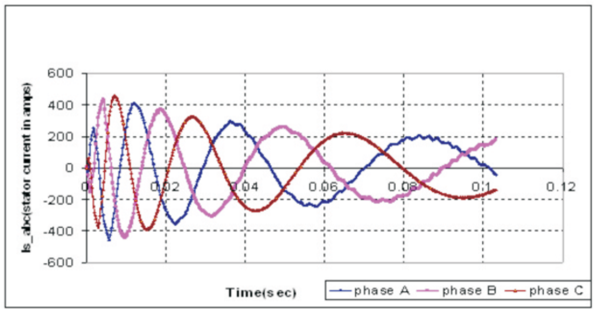


Fig. 18. (c) Stator current of IVCIM using HBPWM controller with $k = 0.003$

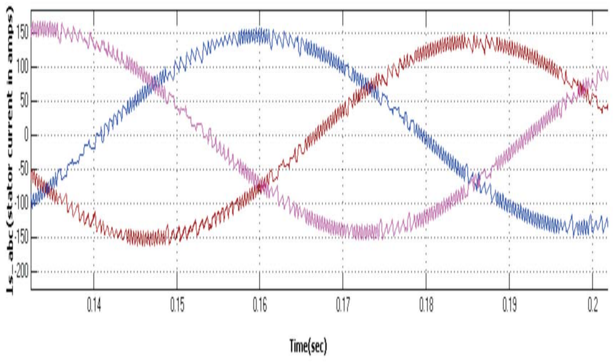


Fig. 18. (d) Zoomed portion of stator current of IVCIM using HBPWM controller with k = 0.003

From the Figs 18(a) and 18(c) it is observed that if the input to the current controller is normalized by a normalization factor of 0.003, the input current to the motor using PDHBPWM controller is smooth but the input current to the motor using HBPWM controller is distorted. The Figs 18(b) and 18(d) show the zoomed portions of the stator currents in the Figs 18(a) and 18(c) respectively. Also the torque pulsations for the motor developed torque are smaller for the drive employing PDHBPWM controller.

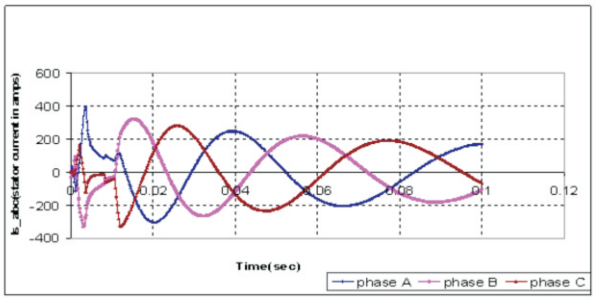


Fig. 19. (a) Stator current of IVCIM using PDHBPWM when phase-A current error is zero

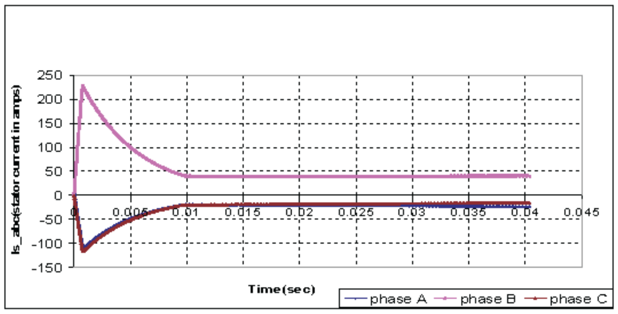


Fig. 19. (b) Stator current of IVCIM using HBPWM when phase-A current error is zero

The Figs 19(a) and 19(b) show the fault tolerance property of neural networks. When any one of the inputs to the HBPWM controller becomes zero, the output current of the inverter which is the input current to the stator of the IVCIM becomes zero. Hence the IVCIM drive employing the HBPWM controller ceases to work in such a case unlike the PDHBPWM controller. In the PDHBPWM controller the other two inputs compensate the lack of the non existing input and hence the drive still works. The stator currents waveforms shown in Figs 19(a) and 19(b) are obtained by simulating both the drives when the A-phase current error input to the controllers is zero.

Hysterisis Band Width	%THD of stator current (phase - A) of IVCIM using	
	HBPWM controller	PDHBPWM controller
0.02	3.679	3.637
0.5	4.210	3.0375

Switching loss

The switching loss of the inverter using PDHBPWM controller is decreased compared to that of the inverter using HBPWM controller in the IVCIM drive. This is proved by comparing in the both cases for different values of k. For normalization factor $k = 1$, α for the drive using HBPWM controller is 0.26 α for the drive using PDHBPWM controller is 0.20 For normalization factor $k = 0.1$, α for the drive using HBPWM controller is 0.64 α for the drive using PDHBPWM controller is 0.32 *For normalization factor* $k = 0.003$, α for the drive using HBPWM controller is 0.96 α for the drive using PDHBPWM controller is 0.40 We can observe that in all the above cases, α is smaller for the drive employing PDHBPWM controller. Hence the inverter switching losses are smaller when PDHBPWM controller is used instead of conventional HBPWM controller.

Total Harmonic Distortion

The % total harmonic distortion (%THD) of the input current to the stator of the IVCIM using HBPWM controller and PDHBPWM controller respectively for hysterisis band widths of 0.02

Table 2: Comparison of %THD

Hysterisis Band Width	%THD of stator current (phase - A) of IVCIM using	
	HBPWM controller	PDHBPWM controller
0.02	3.679	3.637
0.5	4.210	3.0375

and 0.5 are shown in the Table 2:

It can be inferred from the data given in the Table 1 that the use of a PDHBPWM controller reduces the THD of the stator current of the IVCIM. For a hysterisis band width of 0.02 there is no appreciable decrease in the THD but for a hysterisis band width of 0.5 the THD is decreased by 1.1725%.

VII. SUMMARY AND OBSERVATIONS

In this Paper the design of a PDHBPWM current controller for an inverter control is presented. And its modeling in SIMULINK is discussed. To illustrate the operation of PDHBPWM controller an IVCIM drive using PDHBPWM controller is modeled. The results are discussed and compared with the drive using conventional HBPWM controller. It is observed that the PDHBPWM controller reduces the switching losses of the inverter, is highly fault tolerant and reduces the THD of the input current to the VCIM. The speed, torque responses of the motor of both the drives are identical thus indicating that the performance of both the controllers is same. Due to the parallel processing mechanism, it is expected that the neural networks map the non-linear data in a very short time. But, because the operation of the neural network is simulated in a series sequential computer, the processing speed is not so fast. Neural network chips with parallel processing mechanism ensure fast processing speed.

As long as the entire information is available, the performance of both the HBPWM and PDHBPWM controllers are the same. But if there is any information loss like some misconnections in the network or one of the inputs to the network is zero, then the performance of the PDHBPWM is good compared to the HBPWM controller. This is due to the fault tolerance property of the neural network.

When the normalization factor k of current error inputs to the HBPWM and PDHBPWM current controllers is 1, the stator current input and developed torque of the IM of both the drives are similar. As k is decreased upto an extent of 0.003, the stator current input and developed torque of the IVCIM using PDBPWM controller are smooth compared to that of the IVCIM using conventional HBPWM.

Switching losses of the inverter with PDBPWM current controller is less compared to those of the inverter with conventional HBPWM current controller.

The stator current of IVCIM using PDBPWM controller has a lesser THD. The THD is a measure of the amount of harmonics present in a signal. Thus a lower value of THD implies lower harmonic content and hence good control signal. This feature minimizes the unnecessary heating and torque pulsations in the IM.

REFERENCES

[1] Masao Yano, Shigeru Abe, Eiichi Ohno: "History of Power Electronics for Motor Drives in Japan", *IEEE.*

[2] Jacek M. Zurada: "Introduction to Artificial Neural Systems"-West Publishing Company, 2002.

[3] MATLAB 6.5: Help Documentation-Release Notes 13.

[4] I.Boldea & S.A.Nasar:"Vector control of AC Drives".

[5] Bimal K. Bose: "Modern Power Electronics and Ac Drives"-Pearson Education,

[6] Dr. P. S. Bimbhra: "Power Electronics"-Khanna Publications, 2nd Edition, 1998.

[7] Bor-Ren Lin and Richard G.Hoft: "Power Electronics Inverter Control with Neural Network", in *IEEE-APEC*, (San Diego), pp. 128-134, 1993.

[8] Fumio Harashima, Yuzo Demizu, Seiji Kondo, Hideki Hashinmoto: "Application of Neural Networks to Power Converter Control", *IEEE-IAS*, Ann. Mtg. conf. Record, pp.1086-1091, 1989.

[9] Marian P Kazmierkowski, Dariusz Sobczuk: "Improved Neural Network Current Regulator for VS-PWM Inverters" *IEEE*, pp. 1237-1241, 1994

Madichetty Sreedhar, presently doing M.Tech at KIIT University, Bhubaneswar,along with the research is continuing as teaching Assistant under the guidance of Senior Professor Sri A.Das Gupta, Dean School Of Electrical Engineering.

Prasanta Kumar Prusty, presently working as a Assistant Engineer in ORISSA POWER TRANSMISSION CO.OP.LTD. Orissa, Doing M.Tech and research in KIIT University, Bhubaneswar.