IDENTIFICATION AND CONTROL OF INDUCTION MOTOR STATOR CURRENTS USING FAST ON-LINE RANDOM TRAINING OF A NEURAL NETWORK

by

Bruce Burton**, Farrukh Kamran*, Ronald G Harley**, Thomas G Habetler*, Martin Brooke*
and Ravi Poddar*

*Georgia Institute of Technology
School of Electrical and Computer Engr.
Atlanta, GA 30332
Tel: (404)-853-9829
Fax: (404)-853-9171
E-mail: thabetler@ee.gatech.edu

**Dept. of Electrical Engineering
University of Natal
King George V Ave
Durban, 4001 South Africa
Tel: 27-31-260-2725
Fax: 27-31-260-1300

Abstract – Artificial Neural Networks (ANNs) which have no off-line pre-training, can be trained continually on-line to identify an inverter fed induction motor and control its stator currents. Due to the small time constants of the motor circuits, the time to complete one training cycle has to be extremely small. This paper proposes and evaluates a new, fast, on-line training algorithm which is based on the method of random search training, termed the Random Weight Change (RWC) algorithm. Simulation results show that RWC training of an ANN yields performance very much the same as conventional backpropagation training. Unlike backpropagation, however, the RWC method can be implemented in mixed digital/analog hardware, and still have a sufficiently small training cycle time.. The paper also proposes a VLSI implementation which one training cycle in as little as 8 μsec. Such a fast ANN can identify and control the motor currents within a few milliseconds and thus provide self-tuning of the drive while the ANN has no prior information whatsoever of the connected inverter and motor.

INTRODUCTION

The induction motor is a nonlinear system whose parameters vary with time and operating conditions. For high performance applications, such as vector control and direct self control, it is necessary for the controller design to be based on observers and estimation techniques[1], which depend on a simplified model of the motor. Artificial Neural Networks (ANNs) provide an alternative method of observing the input/output relationships of the motor. A previously presented scheme [2] proposed that an ANN be trained *off-line*, i.e., with data obtained apriori to mimic existing stator current controllers. Once sufficiently well-trained, the ANN could replace the original current controller with the advantage of increased speed of execution and fault tolerance. With this approach no further training of the network is possible, after the drive is commissioned. Therefore, the performance of such an off-line trained ANN approach depends upon the amount and quality of training data used which in turn depends on system complexity and the range of operating conditions involved and is also sensitive to parameter variations.

Since the induction machine is a deterministic system for which the equations are well-known, training of an ANN from a random initial condition is not necessarily required. Reference [3] proposes a current regulator for induction machines which maps the electrical equivalent circuit equations onto a feedforward neural network and does not require training. However, like the off-line trained ANN scheme [2], the approach in [3] is also prone to degraded performance because of parameter variations. In order to account for unknown parameter variations, an observer-based scheme, like [1] may be used, but unmodeled non-linearities, such as magnetic saturation, can only be accounted for using an adaptive non-linear model based controller or by using an ANN which is trained while the drive controller (including the ANN) is operating *on-line*. Such an ANN scheme was proposed [4] which used continual on-line training (with no off-line training) to identify and adaptively control the currents and therefore the torque, and thus the speed of an induction machine.. Simulated results showed that this scheme could produce high dynamic performance similar to that achieved with conventional vector control. With an on-line trained scheme, custom tailoring of the ANN architecture and weights to match the structure of the motor equations, although possible, is not necessary.

The scheme in [4] incorporates 3 ANNs that are on-line trained using backpropagation, with two different rates of

0-7803-3008-0/95 $4.00 © 1995 IEEE

execution; a relatively slow rate for the 2 ANNs which accomplishes the rotor speed identification and control functions, and a much faster rate for the ANN performing the stator current regulation. The stator current loop control scheme illustrated in Fig. 1, must run at a high enough rate to cope with the comparatively small electrical time constants of the machine. In order to achieve on-line training at a reasonable sampling frequency, e.g. 10 kHz, one epoch (one ANN weights update cycle based on the training error) needs to be completed in approximately 50μs. Due to the lack of a suitable ANN ASIC, the adaptive on-line trained current controller ANN of [4] was implemented in software [5,6] on a transputer, but the sampling frequency was limited by the ANN computation overhead to approximately 500 Hz.

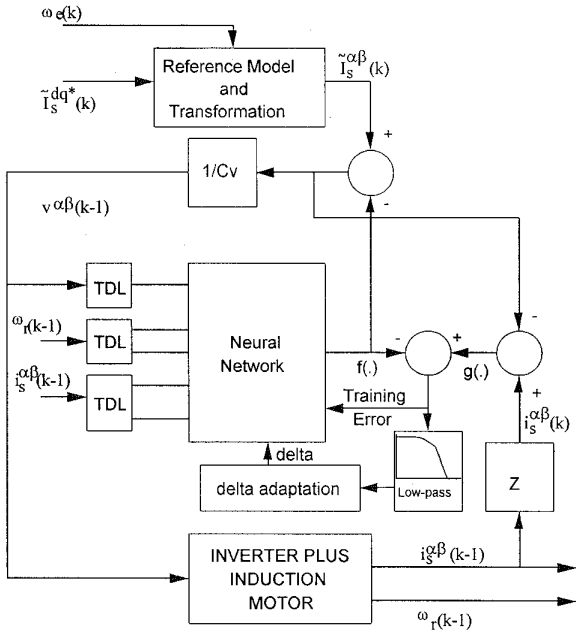

Fig. 1. Adaptive ANN stator current controller block diagram.

This paper proposes and investigates the use of a new fast on-line training algorithm for feedforward ANNs suitable for hardware implementation and which can meet the timing constraints described above. As opposed to backpropagation, this new method called *Random Weight Change* (RWC) training, is robust and insensitive to the non-idealities of analog VLSI circuits.

COMMERCIALLY AVAILABLE ANN HARDWARE

Recently, a zero instruction set computer chip, ZISC036 was introduced by IBM that uses radial basis functions (RBF) as the training method [7]; a similar but more powerful chip using RBF for training has also been developed by Intel and Nestor Inc. [8]. The radial basis function neural networks are not as powerful as the feedforward ones when compared in terms of extrapolation and generalization capabilities. Another possibility explored by some researchers is to combine a feedforward ANN ASIC like the Intel Electrically Trainable Analog Neural Network (ETANN) with external high speed processors to implement the backpropagation [9]. The forward pass is carried out in the ETANN and the weight update computation is done on the external processor; although this method can achieve considerably higher speeds than a software implementation of the ANN, the achieved speed is still too slow for the continual on-line training requirement of the motor application considered in this paper.

A complete VLSI implementation of the feedforward neural net using backpropagation or one of its variants for continual on-line training has not been accomplished to date. The major obstacle in this regard is the sensitivity of these gradient descent training algorithms to the non-linearities and offsets present in hardware analog multipliers and adders. The all-digital implementation takes up much larger chip areas than the analog ones and therefore, a fully *parallel* digital implementation can be realized only for small networks and with lesser number of bits. Backpropagation is sensitive to the bit resolution and fails to converge if the resolution is inadequate [10]. If, instead, the computation is carried out *serially* in digital circuits, learning speed must be sacrificed. Analog implementation of weight circuits and multipliers, on the other hand, has the advantage of fast operating speed and small chip areas, therefore allowing larger circuits to be realized. However, the non-idealities of these analog circuits, as mentioned above, render the use of backpropagation or its modified versions impractical.

All the abovementioned problems prompted the search, not for faster hardware to implement continuous on-line training using backpropagation, but instead for an alternate training algorithm which is more robust and less complex and, therefore, would take much less time to execute on VLSI. Such an algorithm is described in the next section.

THE RANDOM WEIGHT CHANGE TRAINING ALGORITHM

A new ANN training algorithm called Random Weight Change (RWC) is proposed as a variation of a previously reported method of ANN training based on random search for a minimum on the error surface [10]. For applications with hundreds of weights and weight update times in the micro-seconds, the RWC algorithm can be implemented on hardware which is lower in cost and simpler than commercially available hardware since compact analog mixed signal circuitry can be used to perform the weight updates and the forward-propagation of the network. While the RWC algorithm works well for the induction motor application, it is generic in nature and is potentially useful

for many other applications. During each training cycle or *epoch*, each of the network weights is perturbed by a number which has a fixed magnitude, δ, and a random sign. The ANN output error is computed after the weight change. This error is compared to the value of the previous error before the weight change, and based on this comparison, a decision is taken whether to keep the new weights or not. Keeping the ANN input vector fixed, this process is repeated a number of times (i.e., *trials*) during each epoch, and the final weights at the end of the epoch are chosen to be the ones that result in the smallest error during that epoch. The flow chart in Fig. 2 explains one training cycle or epoch in more details.

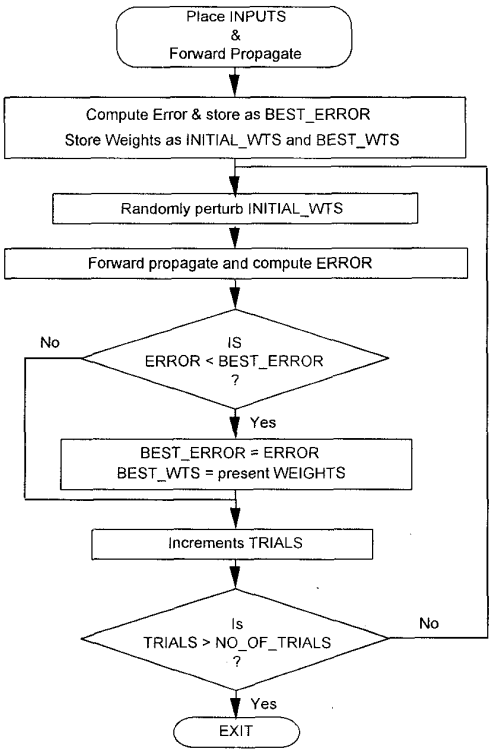

Fig. 2. Flow chart depicting one training epoch of the Random Weight Change training algorithm.

The step size, δ, is a training parameter that needs to be determined heuristically for a specific problem. This is very similar to the gain coefficient, β, in backpropagation and is best thought of as the radius of a hyper-sphere in the n-dimensional error hyper-surface, where n is the total number of network weights. It has been observed during experimentation that the value of δ effects the convergence speed and accuracy and needs to be small, about two orders of magnitude less than the weight magnitudes. As is clear from Fig. 2, each epoch contains N trials, therefore the forward propagation and the random weight change has to be done N times during each epoch. This appears to be a large amount of computation, but, because of the fact that this scheme can be implemented with fully parallel nodes, and the random numbers can be generated very efficiently using shift registers, the all-hardware implementation can achieve very high speeds.

RANDOM WEIGHT CHANGE HARDWARE

A proposed hardware schematic for the RWC algorithm learning ANN is presented in Fig. 3. The hardware is controlled by a conventional micro-controller which generates the clock signals (ph1 and ph2) and several control signals (update, best, rand(i), δ). These control signals can be generated with a low cost single chip micro-controllers. The parallel mixed signal analog-digital hardware carries out all the parallel weight update and forward-propagation operations. It is anticipated that with a 100 MHz digital clock the training epoch, proposed in Fig. 3, would take 800 ns (20 RWC trials). With a less aggressive 10 MHz clock, the epoch would take 8 micro-seconds.

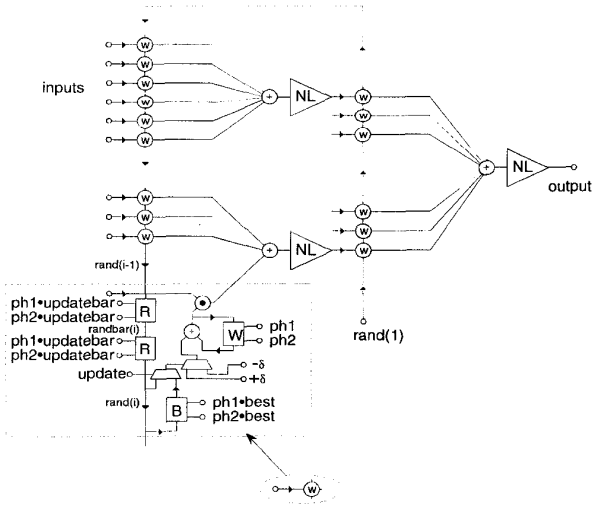

Fig. 3. The Random Weight Change learning hardware schematic.

Each weight circuit in the ANN contains three registers (R, W, and B in Fig. 3) that store the random weight changes, the current weight, and the best weight change for the current epoch, respectively. The random weight changes are shifted into the register, R, and used to update the weight register, W. Then the opposite weight change is shifted into R and used to return W to the original value. The timing of this process is shown in Fig. 4. The step A1 represents the trial of the current weight change, while A2 represents the return to the original value of the weights. The external signal "best" is generated externally from the hardware by calculating the output error and indicates to the hardware that the current weight change is the best so far

among the completed trials in the current epoch. This signal causes the current weight change to be saved in the B (best) register. When one epoch is complete, the update signal is raised and the best weight change is permanently saved in the weight register (W). Referring to Fig. 4, in step B, the external processor computes that the current weight change is the best for the epoch. In step C the value of the register B (best(i)) is updated. In step D the best overall weight change is made permanent.

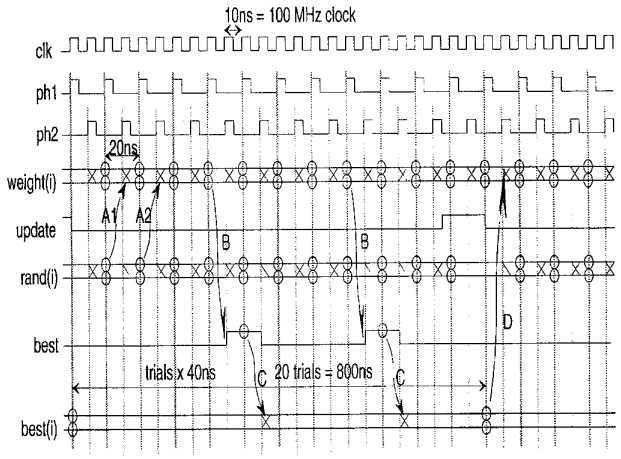

Figure 4. Timing diagram for RWC learning hardware.

RESULTS

In order to assess the suitability of the RWC algorithm to identify and control the motor stator currents, the response of the system in Fig. 1 is computed for several step changes in the magnitude and the frequency of the demanded stator currents. Typical results from several case studies are presented here. The ANN in Fig. 1 consists of 8 inputs and 2 outputs; the number of middle layer neurons, *m,* is varied from one case study to the next. The middle layer outputs pass through sigmoidal non-linearities while the final outputs are linear. The equations for the system in Fig. 1 were given in [5, 6] and are not presented here. During each epoch, the process in Fig. 3 is repeated for the number of trials, N=20, while keeping the inputs fixed.

Case Study One: RWC compared with backpropagation

In order to compare the convergence properties of the RWC and the Backpropagation algorithms, the simulation of the response of the system in Fig. 1 is repeated for both algorithms. Fig. 5 shows the desired current and the actual current, firstly using RWC (Fig. 5(a)) and then using Backpropagation (Fig. 5(b)). No pre-training of the ANN takes place and in each of Figs. 5(a) and (b), the ANN is allowed to identify and control the current from a random initial set of weights.

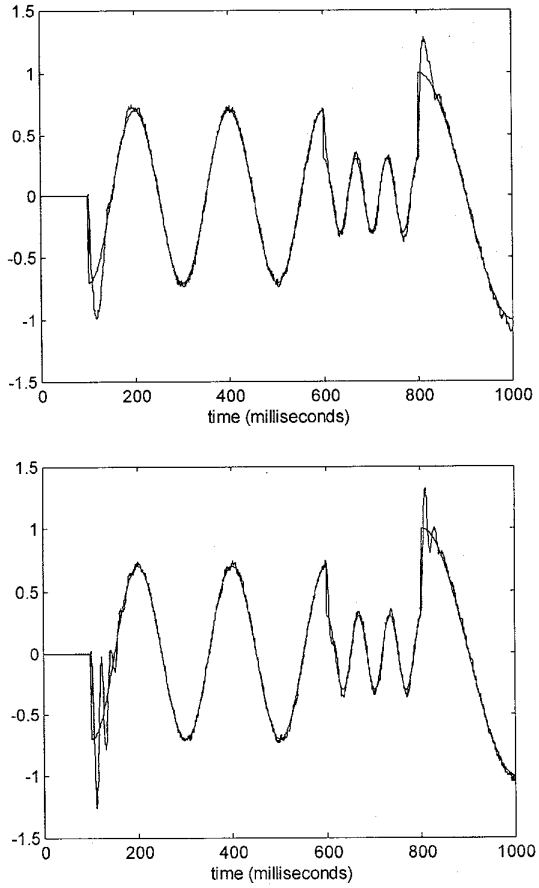

Figure 5. Desired and actual current (d-axis). (a) Using RWC training. (b) Using backpropagation.

From t=0 , initial conditions prevail, but the desired current magnitude and frequency are both set to zero and the ANN lies dormant with initial weights. On-line training, current identification and control then all commence at t=100 ms when the first of three step changes in current is demanded.

step 1:	100ms<t<600 ms,	ω=30 rad/s,	I=0.7 pu,
step 2:	600ms<t<800 ms,	ω=90 rad/s,	I=0.3 pu
step 3:	t >800 ms,	ω=15 rad/s,	I=1.0 pu

The particular sequence and nature of the step changes do not represent any particular mode of induction motor operation, but are the same as the sequence used by [4] to illustrate that convergence occurs each time following any rapid change.

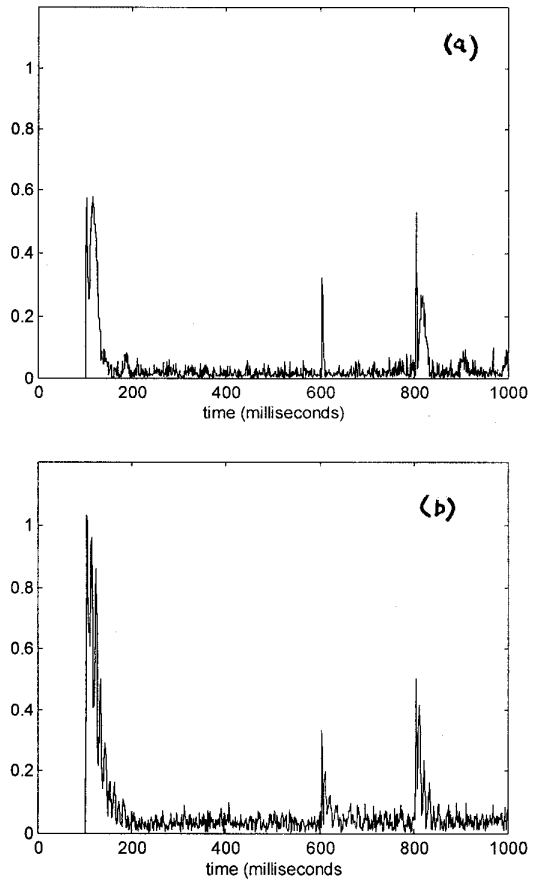

Figure 6. The *training error* used to make weight updates. (a) For the RWC method. (b) For the backpropagation.

Measurement noise is artificially introduced into the simulations in order to more closely approximate the real system, but the PWM and the inverter are not modeled. The results in Fig. 5 show that both training methods allow the ANN to quickly identify the stator currents from a random initial state and thereafter closely track all three steps of desired response. The RWC results of Fig. 5(a) are at least as good as those of backpropagation in Fig. 5(b). The *training error* (which is a variable in Fig. 1) for each method is shown in Fig. 6 and confirms that RWC, in fact, converges at least as well as backpropagation, although neither method has been optimized for these results.

Case Study Two: Convergence of RWC

Both RWC and Backpropagation in Case Study One started with a random initial set of weights. In order to prove that the RWC convergence does not depend on the values of the initial set, the RWC simulation of Case Study One is repeated nine more times, each time starting with a different random set of weights. This yields ten sets of curves like the ones in Figs. 6(a and b), but with each one differing slightly from the other due to the random nature of the RWC algorithm. Rather than overlaying ten such curves, only a single curve which represents the average of the ten training errors is shown in Fig. 7. The plot clearly shows an average convergence as good as the single result of Fig. 6(a). This means that the good result in Fig. 5(a) is not the consequence of a "lucky break" in choosing the initial weights, but that RWC converges each time, irrespective of the initial values.

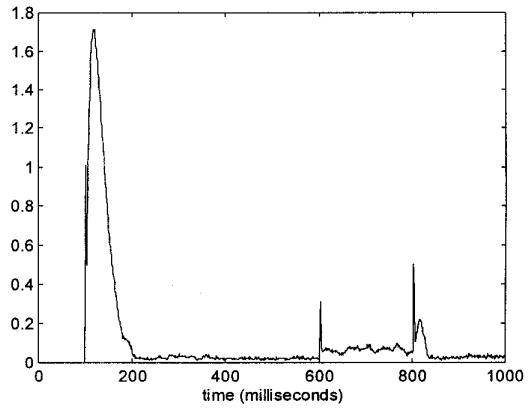

Figure 7. Averaged *training error* over ten independent simulation runs using the RWC method from arbitrary initial conditions.

Case Study Three: Number of Inner Layer Nodes

The ANN in Case Studies One and Two used m=12 inner layer nodes. In order to test whether the convergence properties of the RWC algorithm depend on the value of m, the test of Fig. 5(a) is repeated in Fig. 8 for an ANN with m=12, 20 and 30 nodes respectively. Obviously the initial set of weights for each number of nodes differs from that of the other set, because of the different lengths of the weight vectors. However, the same value of the step size δ=*0.005* is used for all the results in Fig. 8.

The marked difference in the initial responses (from t=100 ms to about 150 ms) of the 12, 20 and 30 node systems is determined maninly by their different random starting weights. After this initial period it appears that the 20 node ANN tracks the desired curve better than the 12 node ANN, and that the 30 node ANN tracks best of all three. Once the desired current settles into a steady sinewave, the differences in tracking ability become almost insignificant. When the second and third transients occur the 20 and 30 node systems seem to track only slightly better than the 12 node system. This is to be expected since a 30 node system contains more high frequency training information than a 12 node system Nevertheless, the

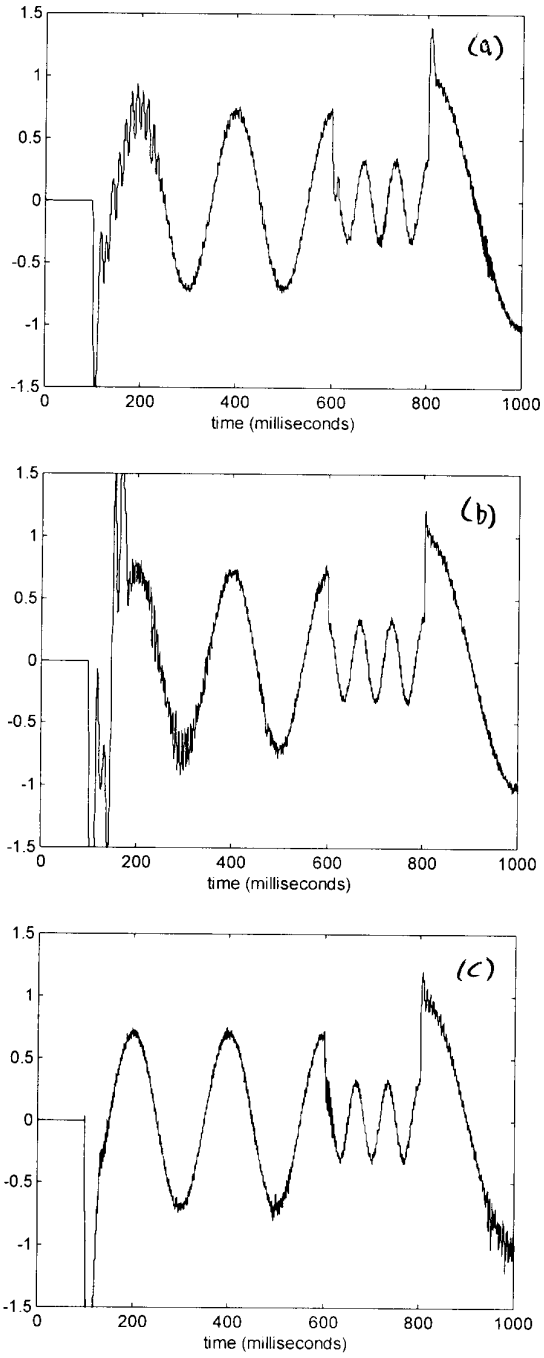

Figure 8. Depicting the effect of number of nodes while using RWC training. (a) Simulation results using 12 nodes. (b) Using 20 nodes and (c) using 30 nodes.

performance of the 12 node system appears acceptable and has the advantage of being a smaller system with a lower computational burden to implement.-

Case Study Four: Influence of Step Size, δ

In the previous case studies the frequency of the current was stepped to 30, then to 90 and then to 15 rad/s. Induction motors often operate at frequencies higher than these and so for the next case study the frequency of the 12 node system is stepped to 314 rad/s (50 Hz) in order to verify that the same system which previously converged (Fig. 8(a)) for the slower frequencies will also converge at a higher frequency. The simulation results appear in Fig. 9 and are repeated for different values of δ, but each simulation now starts with the same random set of initial weights so that the differences in the results are not due to different initial conditions.

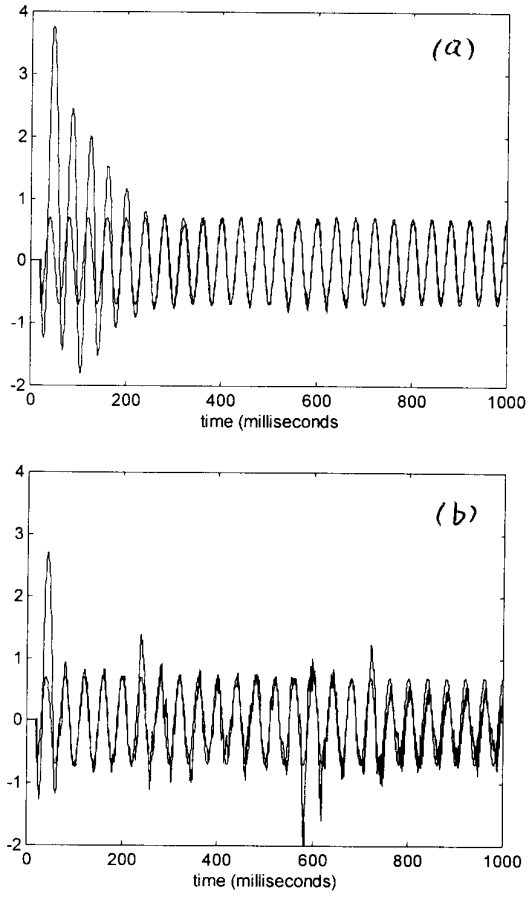

Figure 9. The desired and actual currents showing he effect of δ on learning. (a) δ = 0.005 and (b) δ = 0.02.

The results of Fig. 9(a) (when δ=0.005 which is the same value as for Fig. 8(a)) show that the actual current needs a few cycles to converge to the commanded sinewave. Thereafter the tracking is close although a small steady state error continues to exist at the end. Further simulations with slightly larger values of δ yield a smaller

steady state error but soon result in an oscillatory response; all these results are not shown except for a selected case when δ=0.02 in Fig. 9(b) in order to illustrate the oscillatory response when δ is too large.

The results of Fig. 9 show that the influence of the training step size δ is like the proportional gain of a traditional feedback controller. In other words, small oscillations set in before the steady state error can be driven to zero by increasing δ. This suggests that the step size should not be a fixed value δ, but might also contain a term sensitive to the size of the error.

CONCLUSIONS

Backpropagation has been previously used for identification and control of the stator currents of an induction motor. However, ANN hardware implementation of backpropagation must be capable of executing one epoch in less than 50 μs, a requirement imposed by the continual on-line training and a sampling frequency of 10 kHz. An execution speed of this order, using backpropagation, is not realizable with existing VLSI technology. This paper has propose a new fast on-line Random Weight Change (RWC) training algorithm for feedforward ANNs with a potential for mixed signal (analog/digital) VLSI realizability able to meet the above time constraint. The RWC algorithm is based on the method of random search, is computationally simple and suitable for VLSI implementation; moreover, it produces results comparable to backpropagation.

Results have been presented to show that an ANN with 12 nodes in the inner layer can be trained to identify and control the motor currents almost as well as one with 30 nodes, but the 12 node ANN is preferred because of its lower computational requirement on the implementation hardware. The results also show that the 12 node ANN system performs well both at low and high frequency motor currents. Moreover, the value of the training step size δ influences the system behavior in the same way as the proportion gain of a traditional feedback controller.

An inverter fed adjustable speed induction motor could be identified and its stator currents controlled within a few milliseconds of the startup and thus provide self-commissioning while the ANN has no prior information whatsoever of the inverter and the motor connected to it.

REFERENCES

[1] Atkinson D J, Acarnley P P and Finch J W, " Observers for Induction Motor States and Parameter Estimation," *IEEE Trans. on Industry Applications*, Vol 27, No 6, Nov./Dec. 1991, pp 1119-1127

[2] Buhl M R, and Lorenz R D, "Design and Implementaion of Neural Networks for Digital Current Regulation of Inverter Drives," *Conf. Rec. of IEEE IAS Annual Meeting*, Oct. 1991, Detroit, pp 415-421

[3] Seidl D R, Kaiser D A and Lorenz R D, "One-Step Space Vector PWM Current Regulation Using a Neural Network," *Conf. Rec. of IEEE IAS Annual Meeting*, Oct. 1994, Denver, pp 867-874.

[4] Wishart M T and Harley R G, " Identification and Control of an Induction Machine using Artificial Neural Networks," *Conf. Rec. of IEEE IAS Annual Meeting*, Oct. 1993, Toronto, pp 703-709.

[5] Burton B, Harley R G, Diana G and Rodgerson J R, " Implementation of a Neural Network to Adaptively Identify and Control VSI Fed Induction Motor Stator Currents," *Conf. Rec. of IEEE IAS Annual Meeting*, Oct. 1994, Denver, pp 1733-1740

[6] Burton B and Harley R G, " Reducing the Computational Demands of Continually Online Trained Artificial Neural Networks for System Identification and Control of Fast Processes," *Conf. Rec. of IEEE IAS Annual Meeting*, Oct. 1994, Denver, pp 1836-1843

[7] Å. Eide, Th. Lindblad, C. S. Lindsey, M. Minerskjöld, G. Sekhniaidze and G. Székely, "An implementation of the Zero Instruction Set Computer (ZISC036) on a PC/ISA-bus card," *WNN/FNN*, Washington DC, Dec. 1994

[8] Chin Park, Kenneth Buckmann, Jay Diamond, Umberto Santoni, Siang-Chun The, Mark Holler, Michael Glier, Christopher L. Scofield and Linda Nunez, "A Radial Basis Function Neural Network with On-chip Learning," *Proceeding of the IJCNN*, Oct. 1993, Japan, pp 3035-3038.

[9] Gustavo Cancelo and Sten Hansen, "Analog Neural Network Development System with fast on line training capabilities," *IEEE IECON*, Sep. 1994, Italy, pp 1396-1400.

[10] Kenichi Hirotsu and Martin Brooke, "An Analog Neural Network Chip with Random Weight Change Learning Algorithm," *Proceeding of the IJCNN*, Oct. 1993, Japan, pp 3031-3034.