Neural Network Applications in Power Electronics and Motor Drives—An Introduction and Perspective

Bimal K. Bose, *Life Fellow, IEEE*

Invited Paper

***Abstract*—Artificial intelligence (AI) techniques, particularly the neural networks, are recently having significant impact on power electronics and motor drives. Neural networks have created a new and advancing frontier in power electronics, which is already a complex and multidisciplinary technology that is going through dynamic evolution in the recent years. This paper gives a comprehensive introduction and perspective of neural network applications in the intelligent control and estimation for power electronics and motor drives area. The principal topologies of neural networks that are currently most relevant for applications in power electronics have been reviewed including the detailed description of their properties. Both feedforward and feedback or recurrent architectures have been covered in the description. The application examples that are discussed in this paper include nonlinear function generation, delayless filtering and waveform processing, feedback signal processing of vector drive, space vector PWM of two-level and multilevel inverters, adaptive flux vector estimation, and some of their combination for vector-controlled ac drive. Additional selected applications in the literature are included in the references. From the current trend of the technology, it appears that neural networks will find widespread applications in power electronics and motor drives in future.**

***Index Terms*—Backpropagation network, induction motor drive, intelligent control and estimation, neural network, perceptron, recurrent network, space vector PWM.**

I. Introduction

THE ARTIFICIAL INTELLIGENCE (AI) techniques, such as expert system (ES), fuzzy logic (FL), artificial neural network (ANN or NNW), and genetic algorithm (GA) have recently been applied widely in power electronics and motor drives. The goal of AI is to plant human or natural intelligence in a computer so that a computer can think intelligently like a human being. A system with embedded computational intelligence is often defined as an "intelligent system" that has "learning," "self-organizing," or "self-adapting" capability. Computational intelligence has been debated for a long time, and will possibly be debated for ever. However, there is no denying the fact that computers can have adequate intelligence to help solving our problems that are difficult to solve by traditional methods. Therefore, it is true that AI techniques are now being extensively used in industrial process control, image processing, diagnostics, medicine, space technology, and information management system, just to name a few. While ES and FL are rule-based, and tend to emulate the behavioral nature of human brain, the NNW is more generic in nature that tends to emulate the biological neural network directly. The history of NNW goes back to 1940s, but its advancement was camouflaged by the glamorous evolution of modern-day digital computers. From the beginning of 1990s, the NNW technology captivated the attention of a large segment of scientific community. Since then, the technology has been advancing rapidly and its applications are expanding in different areas. The GA theory (also known as evolutionary computation) was proposed in 1970s and it is based on principles of genetics (or Darwin's survival of the fittest theory of evolution). Basically, it solves optimization problem by an evolutionary process resulting in a best (fittest) solution (survivor). Lotfy Zadeh, the inventor of FL, defined ES as hard or precise computing and FL, NNW, and GA as soft or approximate computing.

Among all the branches of AI, the NNW seems to have maximum impact on power electronics and motor drives area that is evident by the publications in the literature. However, the literature in this area is hardly more than 10–15 years old. In fact, the NNW technology itself is advancing fast in the recent years and its applications in different areas are expanding rapidly. Modern advanced intelligent systems often tend to hybrid neuro, fuzzy, and GA techniques for improvement of performance. The main body of this paper will describe principles of different NNWs and their applications in power electronics and motor drives.

II. Structure of Neuron

A. Biological and Artificial Neurons

A NNW consists of a number of artificial neurons that are interconnected together. The structure of artificial neuron is inspired by the concept of biological neuron shown in Fig. 1(a). Basically, it is the processing element in the nervous system of the brain that receives and combines signals from other similar neurons through thousands of input paths called dendrites. Each input signal (electrical in nature), flowing through dendrite, passes through a synapse or synaptic junction as shown. The junction is an infinitesimal gap in the dendrite which is filled with neurotransmitter fluid that either accelerates or retards the flow of the signal. These signals are accumulated in the nucleus (or soma), nonlinearly modified at the output before flowing to

Manuscript received May 15, 2006; revised September 9, 2006. Abstract published on the Internet November 30, 2006.

The author is with the Department of Electrical and Computer Engineering, University of Tennessee, Knoxville, TN 37996-2100 USA (e-mail: bbose@utk.edu).

Digital Object Identifier 10.1109/TIE.2006.888683

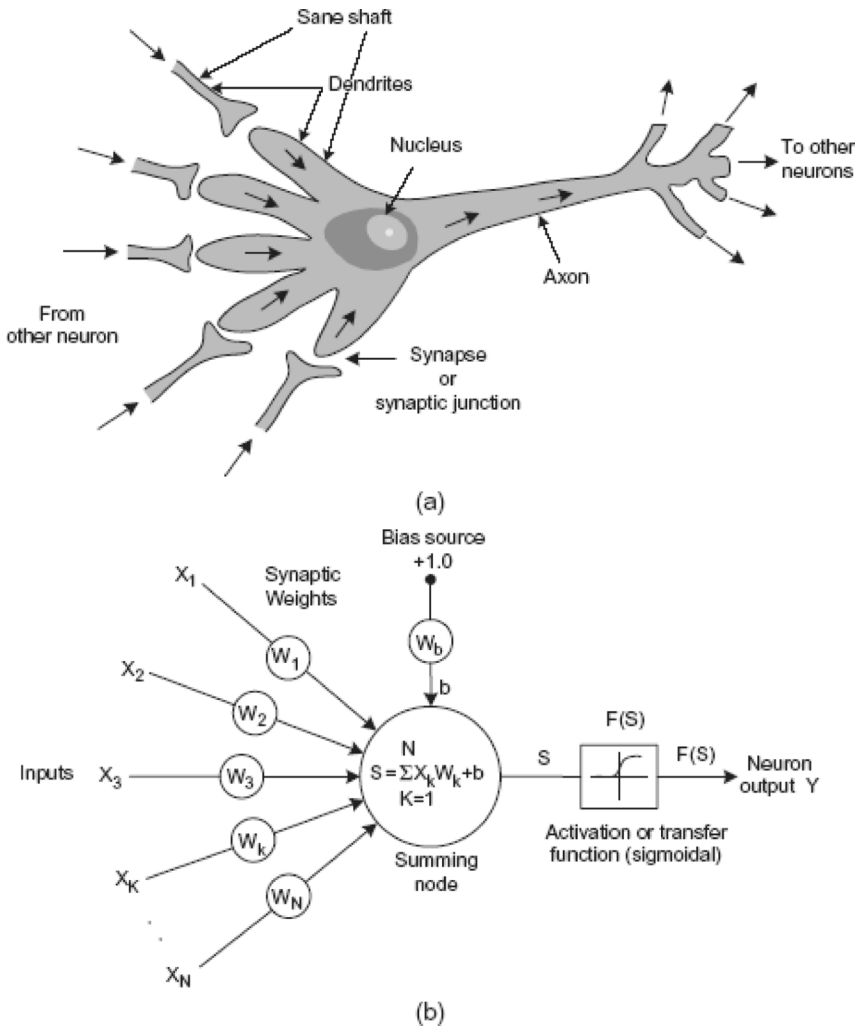

Fig. 1. (a) Structure of biological neuron. (b) Model of artificial neuron.

other neurons through the branches of axon as shown. The adjustment of the impedance or conductance of the synaptic gap by the neurotransmitter fluid contributes to the "memory" or "intelligence" of the brain. According to the theory of the neuron, we are led to believe that our brain has distributed associative memory or intelligence characteristics which are contributed by the synaptic junctions of the cells. It may be interesting to note here that when a human baby is born, it has around 100 billion neurons in the brain. Typically, from the age 40, around one million neurons die every day.

The model of an artificial neuron that closely matches the biological neuron is shown in Fig. 1(b). Basically, it has op-amp summer-like structure. The artificial neuron (or simply neuron) is also called processing element (PE), neurode, node, or cell. Each input signal (continuous variable or discrete pulses) flows through a gain or weight (called synaptic weight or connection strength) which can be positive (excitory) or negative (inhibitory), integer or noninteger. The summing node accumulates all the input-weighted signals, adds to the weighted bias signal b, and then passes to the output through the nonlinear (or linear) activation or transfer function (TF) as shown.

1) Activation Functions: Several common type activation functions used in artificial neuron are shown in Fig. 2. These are defined, respectively, as linear (bipolar), threshold, signum, sigmoidal (or log-sigmoid), and hyperbolic tan (or tan-sigmoid). Another type of function that is often used is the Gaussian function, but is not included here. The magnitude of these functions varies between 0 and 1, or -1 to $+1$ as indicated. The linear function can be unipolar or bipolar. With slope of infinity, it transforms to threshold or signum function, respectively. The sigmoidal and hyperbolic tan functions are commonly used in power electronic system. Their mathematical expressions are included in the respective figure, where α is the gain or coefficient that adjusts the slope or sensitivity. Both these functions are differentiable, and the derivative dF/dS is maximum at $S = 0$. All these functions are squashing in nature, i.e., they limit the neuron response between the asymptotic values. Note that nonlinear activation function contributes to the nonlinear transfer characteristics of neuron that permit nonlinear input–output mapping of NNW which will be discussed later. However, with linear activation function, this nonlinearity is lost.

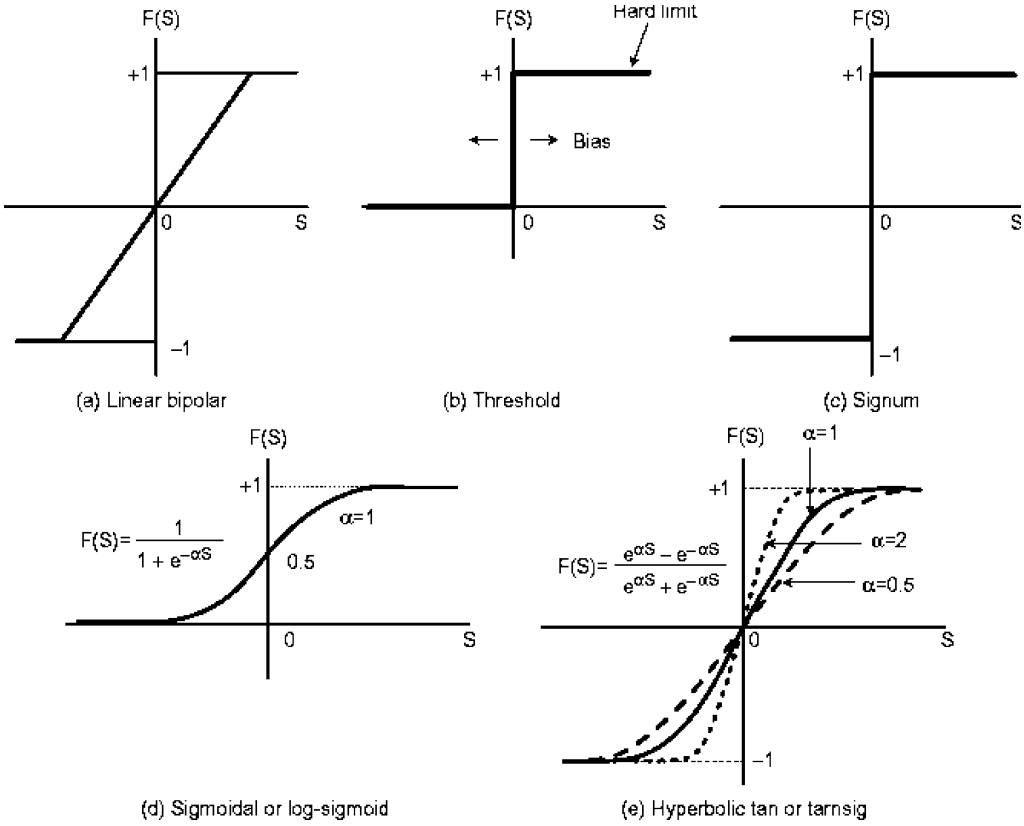

Fig. 2. Several activation functions of artificial neuron.

III. Structures of Artificial Neural Network

A. Network Models

The interconnection of artificial neurons results in NNW (often called neurocomputer or connectionist system in literature), and its objective is to emulate the function of a human brain in a certain domain to solve scientific, engineering, and many other real-life problems. The structure of biological neural network is not well-understood, and therefore, many NNW models have been proposed. A few NNW models can be listed from the literature [1], [2] as follows.

1) Perceptron
2) Adaline and Madaline
3) Backpropagation (BP) Network
4) Radial Basis Function Network (RBFN)
5) Modular Neural Network (MNN)
6) Learning Vector Quantization (LVQ) Network
7) Fuzzy Neural Network (FNN)
8) Kohonen's Self-Organizing Feature Map (SOFM)
9) Adaptive Resonance Theory (ART) Network
10) Real-Time Recurrent Network
11) Elman Network
12) Hopfield Network
13) Boltzmann Machine
14) Recirculation Network
15) Brain-State-In-A-Box (BSB)
16) Bi-Directional Associative Memory (BAM) Network

Generally, NNWs can be classified as feedforward and feedback or recurrent types. In the feedforward class, the signals flow only in the forward direction (see Figs. 3 and 4), whereas in recurrent neural network (RNN), the signals can flow in forward as well as backward or lateral direction (see Fig. 5). A network can be defined as static or dynamic, depending on whether it emulates static or dynamic system. A NNW is characterized by input–output mapping property. For static mapping, the feedforward NNWs are important, whereas for dynamic or temporal mapping, the RNNs are important [1]. The detailed description of all the NNW topologies is beyond the scope of this paper. Only a few topologies that are most relevant for power electronic systems will be discussed. Currently, around 90% of NNW applications use feedforward architecture, particularly the backpropagation network is most popular [1]. Therefore, description of this topology and its application will be emphasized in this paper.

B. Perceptron Network

A perceptron is a simple type of single-layer feedforward NNW which is used for classification of input signal patterns that are linearly separable. It was invented by Frank Rosenblatt in late 1950s [1]. The general structure of perceptron network (multiple perceptrons) with S neurons that use hard-limit unipolar (or threshold) TF is shown in Fig. 3(a). The general input signal vector p has R elements which is multiplied by the weight matrix W *(RXS)* before summation. An element of W is given by $W_{i,j}$, where the weight connects j input element with the ith neuron. The resulting Wp vector is added with the bias vector b to constitute the vector $n = Wp + b$. Each bias signal of a perceptron is generated from a source $+1$

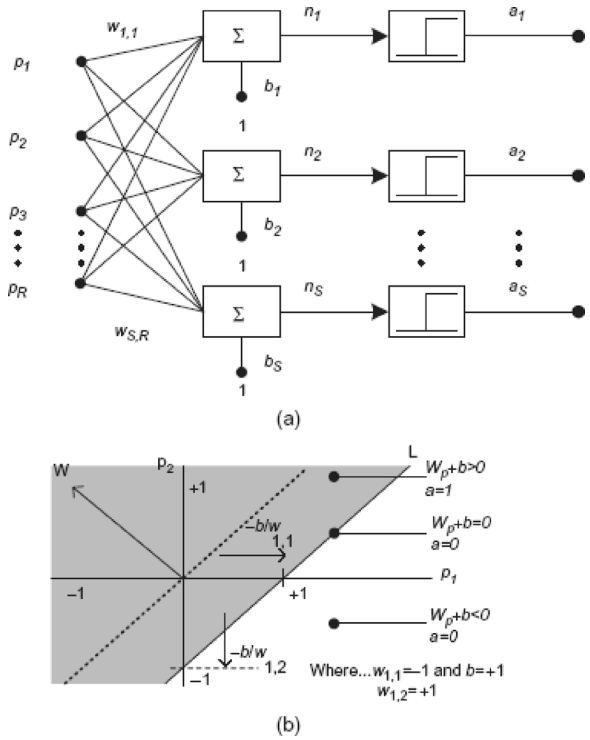

Fig. 3. (a) Perceptron network. (b) Illustration of pattern classification boundary for the upper perceptron.

through a weight as shown. The output vector a is then given by $a = \text{hardlim}(Wp + b)$. The weight is normally indicated by a dot in each link, but is not shown here. The pattern classification property for the upper perceptron only is explained in Fig. 3(b) for p_1 and p_2 inputs, where $w_{1,1} = -1$ and $w_{1,2} = +1$. The shaded area in the figure classifies the p_1 and p_2 inputs that give $+1$ output, whereas the unshaded area gives 0 output. The "decision" boundary line is denoted by L. The bias shifts the boundary line horizontally with the slope remaining the same, whereas the input weights alter its slope. For more than two inputs, the classification boundary is given by a hyperplane. The boundary hyperplane for each perceptron can be designed to be different by assigning different weights and biases. A supervised training algorithm (perceptron learning rule) with the help of computer program [2] can design all the weights for the desired classification boundaries. The NNW training will be discussed later. A simple application of perceptron network may be to isolate healthy signals in a power electronic system from the noisy signals and use for control and monitoring. It can also be trained to solve complex Boolean functions [2] with logical input signals and identification of drive system response characteristics.

It may be mentioned here that Adaline and Madaline (multiple Adalines) NNW has the same structure as that of perceptron except the activation function is bipolar linear. The linear NNW can give linear input-output mapping only. Besides using for pattern classification as discussed above, it can be used for linearizing nonlinear functions (linear function approximation) or pattern association.

C. Backpropagation Network

The feedforward multilayer backpropagation topology, shown in Fig. 4, is most commonly used in power electronics and motor drives. The name "backpropagation" comes from the method of supervised training used for the NNW shown by the lower two blocks in the figure. This will be explained later. The network is normally called multilayer perceptron (MLP) type, because of its evolution from the single-layer perceptron discussed before, but its TF can be different from threshold function. The MLP type NNW is very powerful computationally compared with perceptron NNW. It uses error backpropagation supervised training algorithm which was first described by Paul Werbos in 1974. Subsequently, Rumelhart, Hinton, Williams, McClelland, Parker, LeCun, etc., further made contributions to this method [1]. The example NNW shown in the figure, has three input signals (X_1, X_2, X_3), and two output signals (Y_1 and Y_2). The circles represent the neurons which have associated TF (not shown), and the weights are indicated by dots (often omitted). The network shown has three layers: (a) input layer, (b) hidden layer, and (c) output layer. With five neurons in the hidden layer as indicated, it is normally defined as 3–5–2 network. The input layer is nothing but the nodes that distribute the signals to the middle layer. Therefore, the topology is often defined as two-layer network. The bias source is normally coupled to both hidden and output layer neurons, although it is shown here for the hidden layer only for simplicity. Although the network handles continuous variables, the input and output signals can be continuous, logical, or discrete bidirectional. If the I/O signals are bipolar, the hidden layer neurons usually has hyperbolic tan TF and the output layer has bipolar linear TF. On the other hand, for unipolar signals, these TFs can be sigmoidal and unipolar linear, respectively. The signals within the NNW are processed in per unit (pu) magnitude, and therefore, input signals normalization and output signals descaling or denormalization are used as shown. Although, theoretically, three-layer NNW can solve all the mapping problems, occasionally more than three layers are used.

The NNW has analogy with biological neural network, as mentioned before. Like a biological network, where the memory or intelligence is contributed in a distributed manner by the synaptic junctions of neurons, the NNW synaptic weights contribute similar distributed intelligence. This intelligence permits the basic input–output mapping or pattern recognition property of NNW. This is also defined as associative memory by which when one signal pattern is impressed at the input, the corresponding pattern is generated at the output. This pattern generation or pattern recognition is possible by adequate training of the NNW. The discussion on training and application examples given later will make the concepts adequately clear.

One distinct feature of neurocomputation is that it is very fast when massive parallel computation is done with the help of ASIC chip. An ASIC chip can have embedded read-only memory/random-access memory (ROM/RAM), where the weights can be stored and TFs can be implemented by a look-up table. This is unlike digital signal processing (DSPP/microcomputer-based sequential computation which

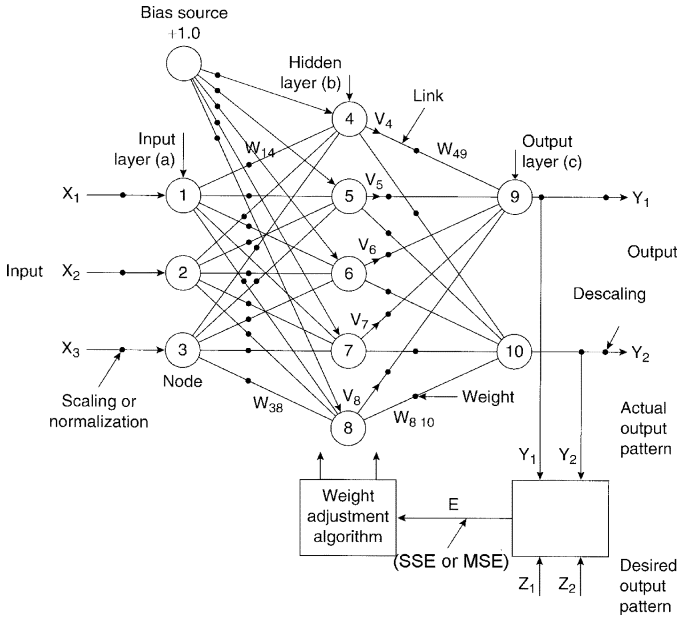

Fig. 4. Three-layer backpropagation network.

is slow. Because of parallel computation, the computation is fault-tolerant, i.e., deterioration of a few weights and/or missing links will not significantly deteriorate the output signal quality. Besides, NNW has noise or harmonic filtering property which will be discussed later.

1) Backpropagation Training: A NNW requires supervised training by example data rather than the traditional programming in a computer system. This is similar to the training of a biological neural network. Backpropagation is the most popular training method for a multilayer feedforward network. In the beginning, input–output example data patterns can be obtained from the experimental results, or from simulation study if mathematical model of the plant is available. Analytical data patterns can also be used. An initial NNW configuration is created with the desired input and output layer neurons dictated by the number of respective signals, a hidden layer with a few neurons, and appropriate TFs. Small random weights are selected so that neuron outputs do not get saturated. With one input pattern, the output is calculated (defined as forward pass) and compared with the desired or target output pattern. With the calculated error E (sum-squared-error (SSE) or mean-squared-error (MSE), as described later), the weights are then altered in the backward direction by backpropagation algorithm until the error between the output pattern and the desired pattern is very small and acceptable (see Fig. 4). A round trip (forward and reverse passes) of calculations is defined as an epoch. Similar training is repeated with all the patterns so that matching occurs for all the patterns. At this point, the network is said to have been trained satisfactorily to perform useful functions. If the error does not converge sufficiently, it may be necessary to increase number of neurons in the hidden layer, and/or add extra hidden layer(s). It has been demonstrated that a three-layer NNW can solve practically all pattern matching problems. Instead of selecting one pattern at a time in sequence, batch method training can be used, where all the patterns are presented simultaneously and final weight updates are made after processing all the patterns.

The weight adjustment for the minimization of error uses the standard gradient descent technique, where the weights are iterated one at a time starting backward from the output layer. Consider that a network is being trained with the input pattern number p, and the squared output error for all the output layer neurons of the network is given by [7]

$$E_p = (d^p - y^p)^2 = \sum_{j=1}^{Q} (d_j^p - y_j^p)^2 \tag{1}$$

where d_j^p = desired output of the jth neuron in the output layer, y_j^p = corresponding actual output, Q = dimension of the output vector, y^p = actual network output vector, and d^p = corresponding desired output vector. The total sum of squared error (SSE) for the set of P patterns is then given by

$$\text{SSE} = E = \sum_{p=1}^{P} E_p = \sum_{p=1}^{P} \sum_{j=1}^{Q} (d_j^p - y_j^p)^2. \tag{2}$$

The weights of the neurons are altered to minimize the value of the objective function SSE by gradient descent method, as mentioned before. The weight update equation is then given as

$$W_{ij}(k+1) = W_{ij}(k) - \eta \left(\frac{\delta E_p}{\delta W_{ij}(k)} \right) \tag{3}$$

where η = learning rate, $W_{ij}(k+1)$ = new weight between ith and jth neurons, and $W_{ij}(k)$ = corresponding old weight. The weights are iteratively updated for all the P training patterns. Sometimes mean square error ($\text{MSE} = \text{SSE}/\text{Q}$) is taken as the objective function. In order to be sure that the SSE converges to a global minimum (i.e., does not get locked up to a local minimum), a momentum term $\mu[W_{ij}(k) - W_{ij}(k-1)]$ is added to the right of (3), where μ is a small value. Further improvement of backpropagation algorithm is possible by making the learning rate step size adaptive, i.e.,

$$\eta(k+1) = u\eta(k) \quad \text{with} \quad u < 1.0 \tag{4}$$

so that oscillation becomes minimum as it settles to the global minimum point. The training is normally done offline because it is time-consuming. A number of backpropagation training methods has been proposed, but the Levenberg–Marquardt (L-M) algorithm is widely used because of its fast convergence. Once the NNW has been trained properly, it should be tested adequately with intermediate data to verify the correct training. Normally, a computer program (such as MATLAB-based Neural Network Toolbox [2] is used for the training.

2) Online Training: In the offline training method, as discussed above, the NNW weights remain fixed or nonadaptive during its operation. In many applications, such as power electronic systems, the NNW has to emulate nonlinear and time-varying functions, where the functions might change depending on the plant operating condition and parameter variation. In such cases, the NNW requires continuous training online. This type of NNW is called adaptive because of online variation of the

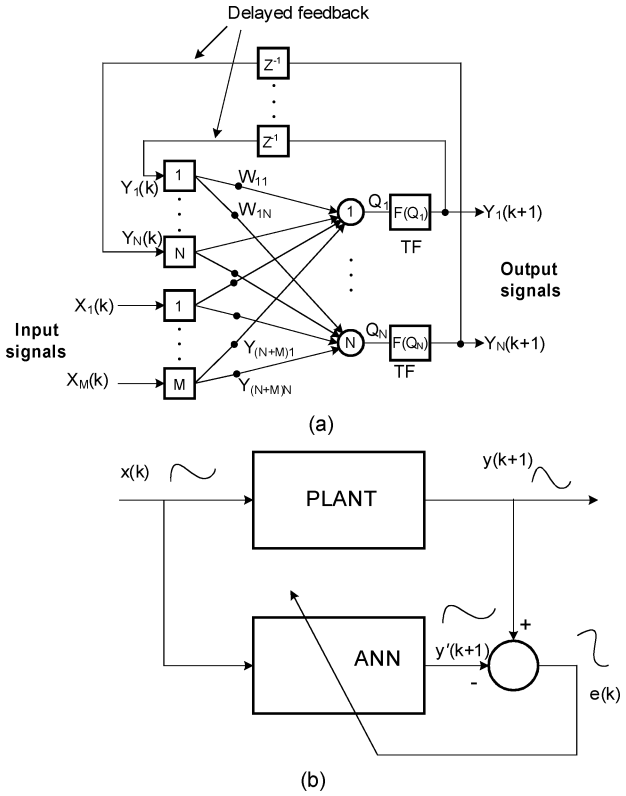

Fig. 5. (a) Structure of real-time recurrent network (RNN). (b) Block diagram for training.

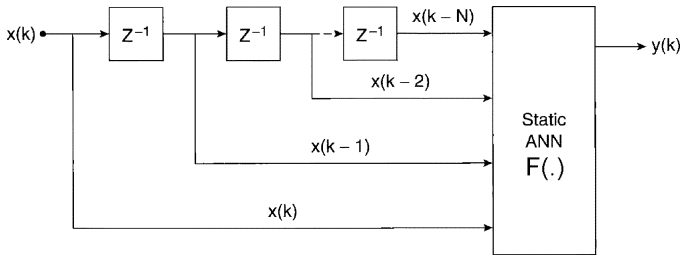

Fig. 6. Time-delayed neural network with tapped delay line.

weights or the structure. Fast and improved version of backpropagation algorithm (such as L-M algorithm) can be used for online training by high-speed PC or DSP to tune the weights, if the process speed is not very fast. Fortunately, DSP speed has improved dramatically and cost of memory has fallen sharply in the recent years. If the range of the plant parameter variation is known ahead of time, the NNW can be trained offline with the nominal parameters, and then the tuning of the weights can be done online by a high-speed DSP. Random weight change (RWC) algorithm [8] was proposed to enhance the speed of online learning. Extended Kalman filter (EKF) training method can also be applied for online training [69].

IV. Neural Networks for Dynamical System

The feedforward NNWs, discussed in the previous section, can give only static input–output nonlinear (or linear) mapping, i.e., the output remains fixed at any instant with the fixed input at that instant. In many applications, a NNW is required to be dynamic; that is, it should be able to emulate a dynamic system with temporal behavior, such as identification of a machine model, control and estimation of flux, speed, etc. Such a network has storage property like that of a capacitor or inductor.

A. Recurrent Network

A recurrent neural network (RNN) normally uses feedback from the output layer to an earlier layer, and is often defined as a feedback network. Fig. 5(a) shows the general structure of a two-layer real-time RNN which was proposed and trained by Williams and Zipser in 1989 [1]. The feedback network with time delays (Z^{-1}) shown can emulate a dynamical system. The output in this case not only depends on the present input, but also prior inputs, thus giving temporal behavior of the network. If, for example, the input is a step function, the response will reverberate in time domain until steady-state condition is reached at the output. The network can emulate nonlinear differential equations that are characteristics of a nonlinear dynamical system. Of course, if TFs of the neurons are linear, it will represent linear system. Such a network can be trained by dynamical backpropagation algorithm [1], where the desired time-domain output from the reference dynamical system (plant) can be used step-by-step to force the ANN (or RNN) output to track by tuning the weights dynamically sample-by-sample, as indicated in Fig. 5(b). As an example, consider a one-input one-output RNN which is desired to emulate a series nonlinear *R-L-C* circuit [plant in Fig. 5(b)]. A step voltage signal $x(k)$ is applied to the plant and the RNN simultaneously. The current response $y(k+1)$ of the plant is the target signal, and it is used to tune the RNN weights. Then, the RNN will emulate the *R-L-C* circuit model. On the same principle, the RNN can emulate a complex dynamical system. An example application in power electronic system for adaptive flux estimation will be discussed later using the EKF algorithm.

B. Time-Delayed Networks

1) Static NNW With Tapped Delay Line Input: Another NNW topology which is popular in dynamical system is the time-delayed neural network (TDNN) shown in Fig. 6. In this case, the single input $x(k)$ is fed to a multi-input static NNW through a tapped delay line which generates the sequence of signals with unit delay (Z^{-1}) as shown. These signals are then multiplied by the respective weights (not shown) and generate the output $y(k)$ through the NNW. The corresponding equation is

$$y(k) = F(.)\left[\sum_{n=0}^{N} W_{nk}x(k-n)\right] \tag{4a}$$

where $F(.)$ represents nonlinear function due to the NNW. The equation represents an Nth order nonlinear dynamical system which is expressed in finite difference form. Note that there is no feedback from the output to the input. The TDNN can be trained by static backpropagation method for the desired dynamical function. Equation (4a) can represent linear dynamical system if the TFs are linear. The network has been used as nonlinear predictor, adaptive transverse filter, and FFT analyzer of a wave [1]. In the last case, for example, sequentially sampled distorted current waves can be presented to a multilayer NNW

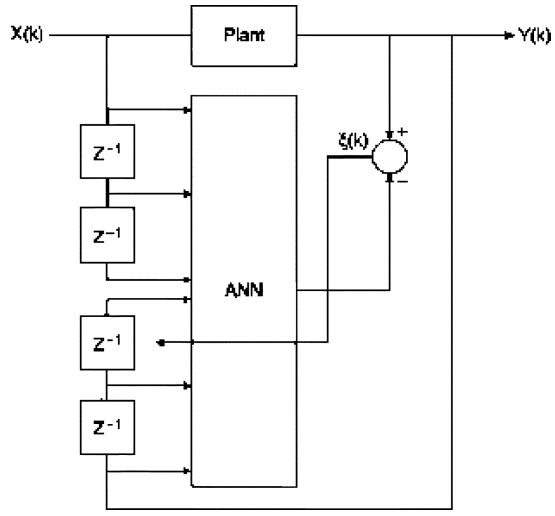

Fig. 7. Neural network with time-delayed input and output.

which can be trained to generate the fundamental amplitude for each wave.

2) Static NNW With Time-Delayed Input and Feedback: The structure of the NNW is shown in Fig. 7, where there are time-delayed inputs, as well as time-delayed outputs as feedback signals. In this case, the NNW is required to emulate the dynamical system given by

$$y(k) = F(.)[x(k), x(k-1), x(k-2), y(k-1), y(k-2)]. \quad (5)$$

The NNW can be trained from the input–output temporal data of the plant by dynamic backpropagation method. As mentioned before, the training data can be generated experimentally from the plant or from simulation result if mathematical plant model is available. If the plant parameters vary, the NNW model generated by offline training method is not valid. In such a case, online training of NNW with adaptive weights is essential. It should be mentioned here that the structure of the NNW will depend on the nature of the dynamical system which is required to be emulated [11].

C. Neural Network Control of Dynamical System

The control of a dynamical system, such as induction motor vector drive, by an AI technique is normally defined as intelligent control. Although all the branches of AI have been used for intelligent control, only NNW-based control [9] will be covered in this section.

1) Inverse Dynamics-Based Adaptive Control: The identification of forward dynamical model of plant has been discussed so far. It is also possible to identify the inverse plant model ($F^{-1}(.)$) by training, as shown in Fig. 8 [9]. In this case, the plant response data $y(k)$ is impressed as the input of the NNW, and its calculated output is compared with the plant input which is the target data. The resulting error $\xi(k)$ trains the network as shown so that $\xi(k)$ falls to the acceptable minimum value. After satisfactory training and testing, the NNW represents the inverse dynamical model of the plant. This NNW-based inverse model can be placed in series as a controller with the actual plant, as shown in Fig. 9, so that the plant forward dynamics is totally

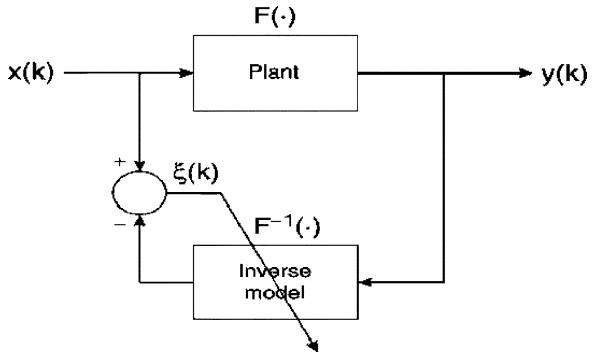

Fig. 8. Training of inverse dynamic model of plant.

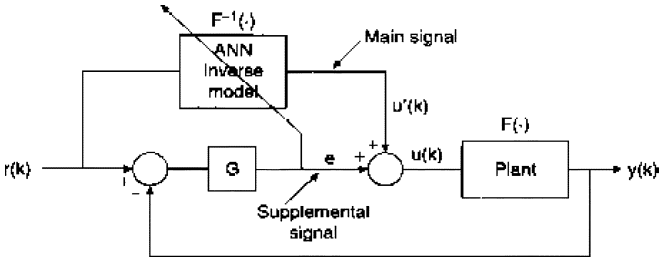

Fig. 9. Inverse dynamic model-based adaptive control of a plant.

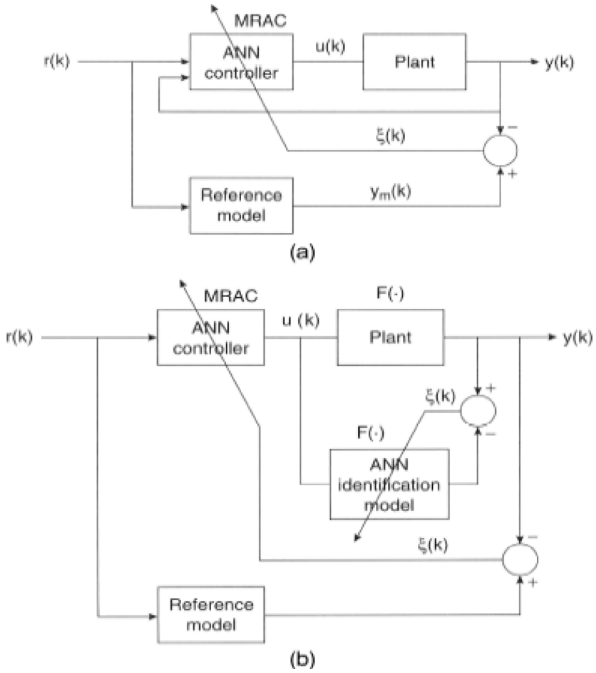

Fig. 10. Model referencing adaptive control by neural network. (a) Direct method. (b) Indirect method.

eliminated, i.e., $F^{-1}(.)F(.) = 1$. Then, ideally the output signals follow the input signals and no feedback control is necessary. However, the actual plant output will deviate from the desired input because of imperfect inverse model and/or plant parameter variation. The feedback control shown generates the supplemental error signal e for the control. The signal e can also be used for online training of the NNW as shown.

2) Model Referencing Adaptive Control (MRAC): The model referencing adaptive control (MRAC) has been widely discussed in the literature. Fig. 10(a) shows the NNW-based

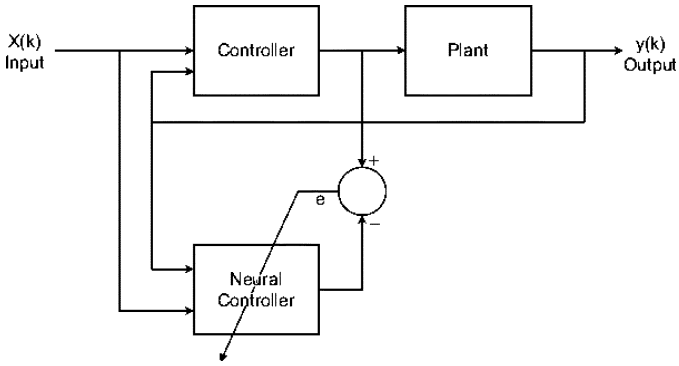

Fig. 11. Training of neural network for emulation of actual controller.

MRAC, where the plant output is desired to track the dynamic response of the reference model. The reference model can be represented by dynamical equation(s) and solved in real-time by a DSP. The error signal $\xi(k)$ between the actual plant output and the reference model output trains the NNW controller online so as to make the tracking error zero. Therefore, the plant with the NNW has the response which is identical to that of the reference model. One problem of this direct method is that the plant lies between the controller and the error, and there is no way to propagate the error backward in the controller by error backpropagation training. This problem is solved in Fig. 10(b) by indirect method. In this case, the NNW identification model $F(.)$ is first generated to emulate the forward model of the plant. This model is then placed in series with the NNW controller (instead of the actual plant) to track the reference model as shown. The tuning of the NNW controller is now convenient through the NNW model.

3) Emulation of Actual Controller: Fig. 11 shows a feedback system with the controller in the front-end. The controller can be simple or complex. It can be static, as well as dynamic. Whatever it is, a neural controller, as shown in the figure, can be trained to emulate the actual controller, and then the NNW can substitute the actual controller. Once the neural controller has been trained and replaced the controller, it can be retrained online for plant parameter variation to make it adaptive. A fuzzy controller, for example, in Fig. 11, that is static and nonlinear, can be *P-I*, *P-D*, or *P-I-D* type. The fuzzy controller can be easily replaced by NNW controller since both controls are based on input–output nonlinear mapping principle. A more complex ANFIS (adaptive neuro-fuzzy inference system) can train the neural controller offline [12] or online by backpropagation method.

V. Neural Network Applications

The history of neural network applications in power electronics and motor drives is very recent, and hardly extends beyond the last 10–15 years, as mentioned before. In this section, several example applications which are mainly based on the author's own research will be discussed briefly because of the length constraint of this paper. The details of the NNW design and their performances for these applications can be found in the respective cited references. There are, of course, many other applications which can be found in the large number of extra references included in this paper.

A. Three-Phase Sine Wave Generation

A simple application of NNW in power electronics is discussed in the beginning to clarify the ideas, particularly the training method. The network, shown in Fig. 12, is designed to generate three-phase sine functions from the input signal X in radians (0 to 2π) [14]. The amplitude A is set to 1 for normalized data handling. The NNW uses a three-layer backpropagation network with the topology of 1–5–3, using tan-sigmoid TF in the hidden layer and linear bipolar TF in the output layer. A bias of $+1$ couples to the hidden and output layer neurons. The bias may not be needed, but it is considered for generality. The training data for X was generated in the step size of 0.1 radian, and the corresponding output target data were generated by MATLAB sine functions. The input-output data file in MATLAB was imported to GUI (graphical user interface) for training. The network was trained by MATLAB Neural Network Toolbox [2] using the L-M algorithm. The reference describes the details of NNW training. After training the network, it was converted to Simulink program for testing. A sawtooth wave with X was impressed at the input and the corresponding three-phase sine waves were generated at the output. The actual output waves were compared with the target waves to validate the success of training. The training was started initially with the topology 1-2-3, and the corresponding test results after 500 epochs of training are shown in Fig. 13. The specified goal of SSE was 0.04, but it failed to converge and the error was locked near 15. The corresponding output waves were highly mismatched. Another round of training was continued with the topology 1-3-3, and the SSE was found to reduce to 0.1 and was locked at this value. Finally, the training was successful with the topology 1-5-3 network where the actual output waves coincided with the target waves and the error converged to the desired goal of 0.001 after 178 epochs, as shown in Fig. 14. The final matrices of weights and biases are summarized below

$$\mathbf{W}^1=\begin{bmatrix} W^1_{11} \\ W^1_{12} \\ W^1_{13} \\ W^1_{14} \\ W^1_{15} \end{bmatrix} = \begin{bmatrix} 0.9746 \\ -1.0423 \\ 0.7937 \\ -0.9571 \\ 0.9504 \end{bmatrix} \mathbf{B}^1 = \begin{bmatrix} B^1_{11} \\ B^1_{12} \\ B^1_{13} \\ B^1_{14} \\ B^1_{15} \end{bmatrix} = \begin{bmatrix} -4.9313 \\ 6.5538 \\ -2.5281 \\ 1.1780 \\ -0.1443 \end{bmatrix}$$

$$\mathbf{W}^2=\begin{bmatrix} W^2_{11} & W^2_{21} & W^2_{31} & W^2_{41} & W^2_{51} \\ W^2_{12} & W^2_{22} & W^2_{32} & W^2_{42} & W^2_{52} \\ W^2_{13} & W^2_{23} & W^2_{33} & W^2_{43} & W^2_{53} \end{bmatrix}$$

$$= \begin{bmatrix} 2.7096 & -4.1398 & -9.5330 & -1.4385 & 5.3972 \\ -8.2493 & -2.1779 & 4.4349 & -7.7484 & -8.3220 \\ 5.5397 & 6.3177 & 5.0981 & 9.1869 & 2.9248 \end{bmatrix}$$

$$\mathbf{B}^2 = \begin{bmatrix} B^2_{11} \\ B^2_{12} \\ B^2_{13} \end{bmatrix} = \begin{bmatrix} 3.4111 \\ 10.8206 \\ -14.2317 \end{bmatrix}.$$

The weights indicate large variation in magnitudes and the importance of good resolution in storage, while implementing the network.

Note that precision three-phase sine functions can be generated by a large look-up table in a DSP. However, NNW has the

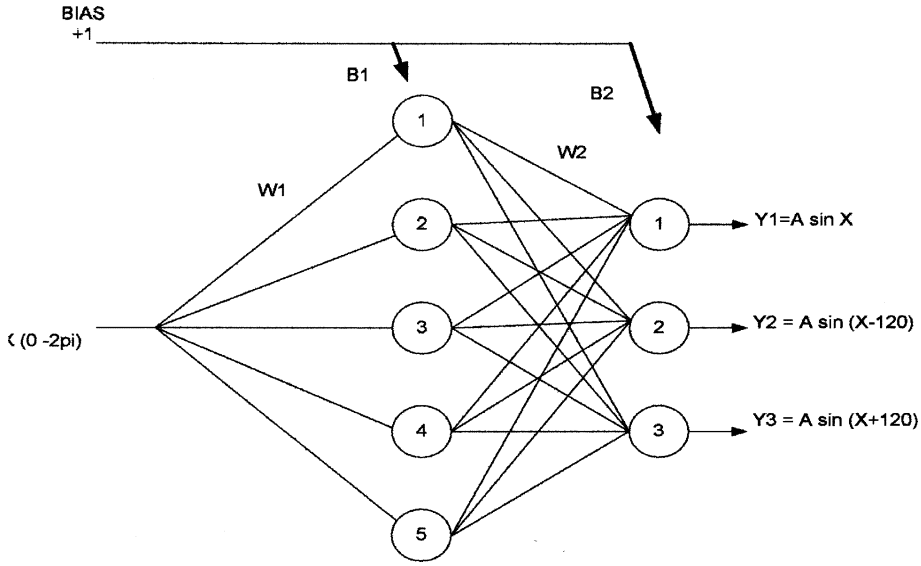

Fig. 12. Backpropagation network for generation of three-phase sine functions.

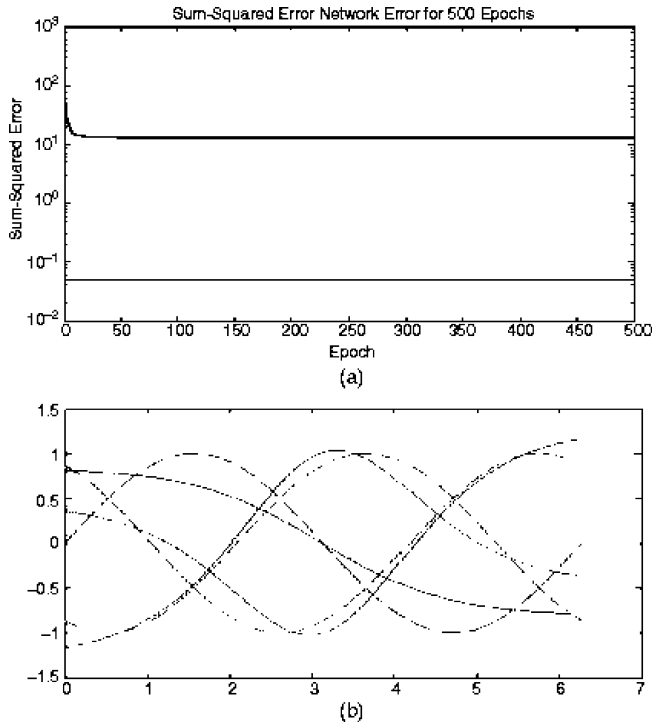

Fig. 13. Training of 1–2–3 network. (a) Training error. (b) Test results.

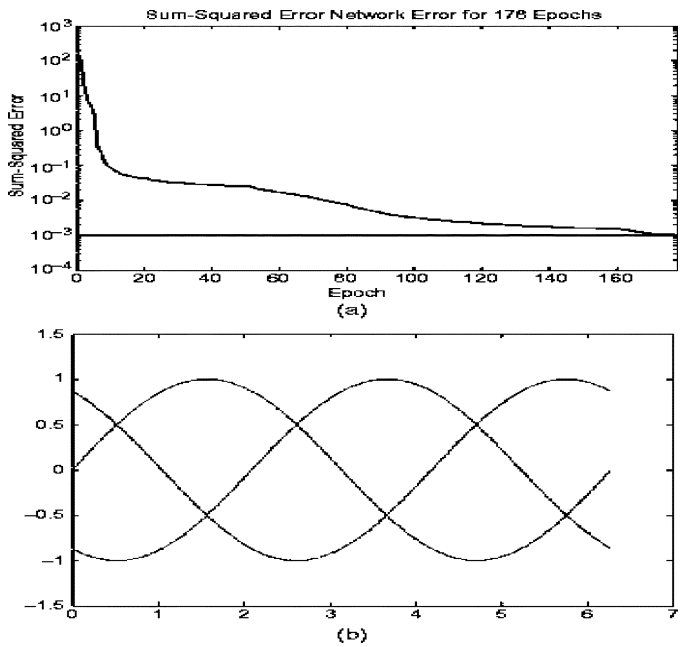

Fig. 14. Training of 1–5–3 network. (a) Training error. (b) Test results.

advantage that it can interpolate with precision the course example data used for the training. Besides, a dedicated ASIC chip can be used for implementation of a task relieving the DSP for other computations.

B. Delayless Filtering and Waveform Processing

1) Single-Phase Square Wave at Constant Frequency: A NNW can be used for delayless filtering of harmonic-rich waveforms in power electronics, as well as waveform processing [17], as shown in Fig. 15. A traditional low-pass filter (LPF) causes frequency-sensitive phase delay and amplitude attenuation at the output. A square-wave line current which is usually encountered for a thyristor or diode rectifier with highly inductive load is considered in this case, although other types of waveforms are also possible. The wave (v_a) and its auxiliary form ($v_a{}'$) through an arbitrary LPF are given as input to the NNW as shown. The output ($v_{af}\angle 0$) comes out as sine wave at the same frequency and proportional to the square-wave amplitude, but is locked at 0° phase angle. The auxiliary wave is needed because with the constant amplitude square-wave input, variable amplitude sine wave output is not possible. Multiphase balanced output waves (in this case three-phase) can be generated easily by programming the NNW. Such waveform

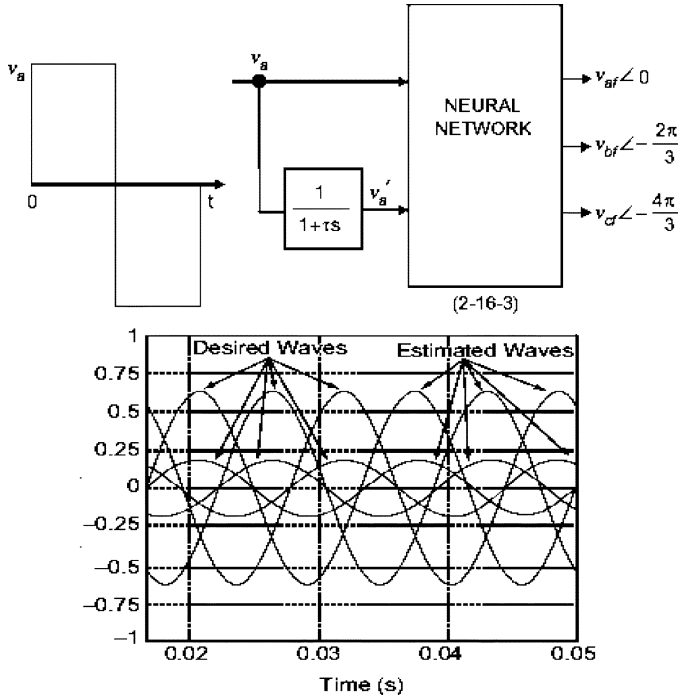

Fig. 15. Single-phase square-wave delayless filtering and multiphasing at output by neural network ($\mathrm{MSE} = 2.33881\mathrm{e}-005$, 600 epochs).

processing may be important for sinusoidal PWM inverter or cycloconverter, where the command can be generated from a simple square-wave signal. The NNW was trained offline with actual v_a and v_a' data at the input and the desired output (generated by prior FFT analysis of the input square-wave). The backpropagation NNW (2–16–3) uses bipolar linear TF at the input and output layers, but hyperbolic tan TF in the hidden layer. The MSE and the number of epochs required for the training are included in the figure title. A small deviation in frequency and distortion of input square-wave will give tolerable error at the output. Generalizing the waveform processing, it can be said that any arbitrary waveform can be transformed to arbitrary output wave at single or multiphase with the help of NNW, and the input–output amplitude tracking relation can be programmed to be linear or nonlinear with arbitrary functional relation. The study also suggests that NNW can be used as zero-crossing detector (ZCD) for a distorted wave.

2) Three-Phase PWM Waves at Variable Frequency: It is possible to have delayless filtering of three-phase PWM waves of a voltage-fed inverter at variable frequency with the help of NNW [17], as shown in Fig. 16. The fundamental voltages at zero phase-shift angle permit correct estimation of feedback signals for vector control of ac drive. The PWM line voltage waves v_{ab}, v_{bc}, and v_{ca} are generated by sinusoidal PWM technique, where only v_{ab} wave is shown. The PWM waves are initially filtered by identical LPFs as shown to convert the discrete waves into continuous waves before processing through the NNW. In Fig. 16(b), the NNW input waves, and the corresponding desired and estimated waves are shown for 10 kHz carrier frequency (f_c), whereas Fig. 16(c) shows results at $f_c = 1.0$ kHz. The MSE and the number of epochs required in the training are included in the figure title. In the later case, the MSE failed to converge below the large value shown. In the later case, the estimated waves indicate more distortion. The test results for 60 Hz fundamental frequency are shown only, although the performance was tested to be satisfactory at variable frequency in the range of 5–60 Hz. Note that in this case the NNW compensates the phase delay and attenuation of LPF (that vary with fundamental frequency) besides filtering the PWM waves. With a more complex structure of NNW and extensive training, the harmonic quality at the output can be improved.

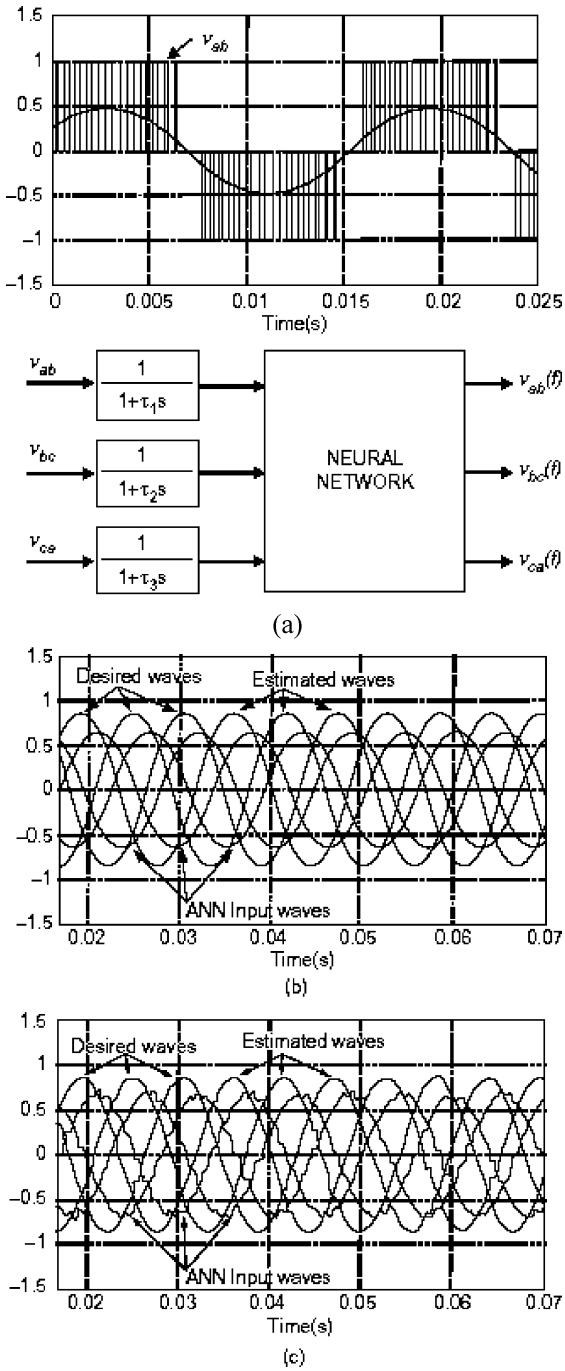

Fig. 16. (a) Three-phase PWM waves delayless filtering by neural network. (b) Network output at $f_c = 10$ kHz ($\mathrm{MSE} = 1.99796e-005$, 90 epochs). (c) Network output at $f_c = 1.0$ kHz ($\mathrm{MSE} = 29.0519e-005$, 500 epochs).

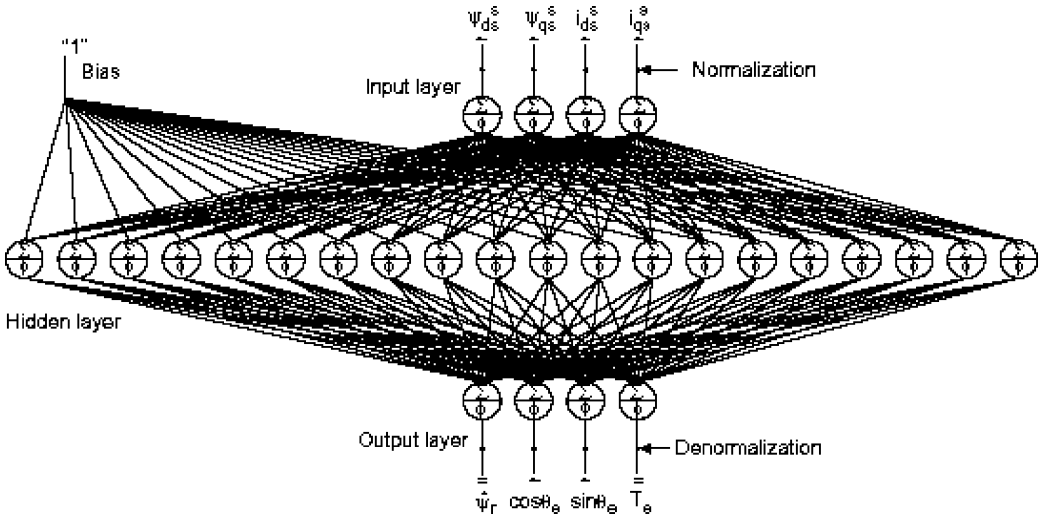

Fig. 17. Neural network for feedback signal estimation.

C. Feedback Signal Estimation of Induction Motor Drive

The vector or field-oriented control of ac drive requires complex calculation of feedback signals. The rotor flux-oriented induction motor vector drive requires computation of the following equations [6] to estimate rotor flux ($\hat{\psi}_r$), torque (T_e), and unit vector ($\cos\theta_e$, $\sin\theta_e$) signals from the machine terminal voltages and currents:

$$v_{qs}^s = \frac{2}{3}v_a - \frac{1}{3}v_b - \frac{1}{3}v_c \tag{6}$$

$$v_{ds}^s = -\frac{1}{\sqrt{3}}v_b + \frac{1}{\sqrt{3}}v_c \tag{7}$$

$$i_{qs}^s = \frac{2}{3}i_a - \frac{1}{3}i_b - \frac{1}{3}i_c \tag{8}$$

$$i_{ds}^s = -\frac{i}{\sqrt{3}}i_b + \frac{1}{\sqrt{3}}i_c \tag{9}$$

$$\psi_{qs}^s = \int (v_{qs}{}^s - i_{qs}^s R_s) \tag{10}$$

$$\psi_{ds}^s = \int (v_{ds}{}^s - i_{ds}^s R_s) \tag{11}$$

$$\psi_{qm}^s = \psi_{qs}^s - i_{qs}^s L_{ls} \tag{12}$$

$$\psi_{dm}^s = \psi_{ds}^s - i_{ds}^s L_{ls} \tag{13}$$

$$\psi_{qr}^s = \psi_{qm}^s \cdot \frac{L_r}{L_m} - i_{qs}^s L_r \tag{14}$$

$$\psi_{dr}^s = \psi_{dm}^s \cdot \frac{L_r}{L_m} - i_{ds}^s L_r \tag{15}$$

$$\hat{\psi}_r = \sqrt{(\psi_{qr}^s)^2 + (\psi_{dr}^s)^2} \tag{16}$$

$$T_e = \frac{3}{2}\left(\frac{P}{2}\right)(\psi_{dr}^s i_{qs}^s - \psi_{qr}^s i_{ds}^s) \tag{17}$$

$$\cos\theta_e = \frac{\psi_{dr}^s}{\hat{\psi}_r} \tag{18}$$

$$\sin\theta_e = \frac{\psi_{qr}^s}{\hat{\psi}_r} \tag{19}$$

where all the standard symbols have been used. Normally, a microprocessor/DSP is used to solve these equations in real-time. A feedforward NNW can also solve the equations except the integrations which are dynamical in nature. The integrations and $3\varphi/2\varphi$ conversions in the front-end can be easily done by op-amp low-pass filters, and the remaining equations can be solved by a backpropagation NNW, as shown in Fig. 17 [18]. Solving (12)–(19) basically involve addition, subtraction, multiplication, division, and square-rooting of signals which are nothing but the process of input–output mapping. Note that all the input signals are in variable frequency and magnitude. The training data of the NNW can be generated by DSP calculation, or simulation result could be used. The three-layer topology (4–20–4) has bipolar linear TF for the input layer, and tan-sigmoid TF for the hidden and output layers (indicated by "ϕ" symbol). The bias is coupled to the hidden layer only (through weights) as indicated. The NNW after training was found to have some harmonic ripple filtering effect. With 15 kHz SPWM of the inverter, the NNW output had very little distortion. However, with 1.0 kHz switching frequency, the distortion was somewhat magnified (less filtering effect).

D. Space Vector PWM of Voltage-Fed Inverter

Space vector modulation (SVM) has recently grown as a very popular PWM method for three-phase voltage-fed inverter with isolated neutral load (such as ac motor) because of its superior harmonic quality and extended undermodulation (UM) range [6] compared with the traditional SPWM method. However, one difficulty of SVM is that it requires complex online DSP-based computation that usually limits its operation up to several kHz of switching frequency. This problem has been solved by applying NNW in SVM implementation. As mentioned before, a feedforward NNW has basically static nonlinear input–output mapping property. The computational delay of this mapping becomes negligible if NNW is implemented by ASIC-based parallel architecture. A feedforward carrier-based PWM technique, such as SVM, can also be looked upon as a nonlinear mapping phenomena, where the command phase voltages can be sampled at the input and the corresponding pulse width patterns can be generated at the output. Therefore, it appears logical

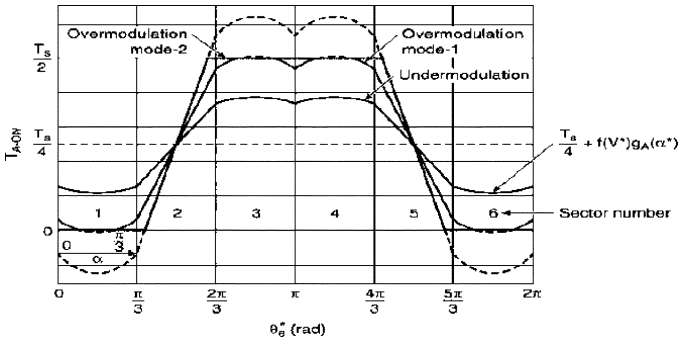

Fig. 18. Turn-on time of phase-A (T_{A-ON}) as function of command vector angle (θ_e) in the six sectors.

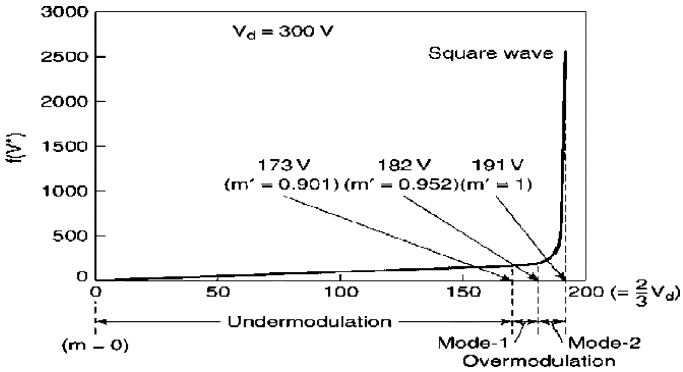

Fig. 19. $f(V^*)$ versus V^* relation in UM and OM regions.

that backpropagation type NNW with high computational capability can implement the SVM algorithm. In addition, the nonlinear voltage transfer characteristics in overmodulation (OM) region can be easily linearized by compensation. The NNW can be conveniently trained offline with the data generated by calculation of the SVM algorithm. Considering the importance of NNW-based SVM, it will be reviewed for two-level, three-level, and five-level inverters.

1) Two-Level Inverter: The NNW-based SVM implementation of a two-level inverter has been discussed in the literature [22]. By prior analysis, it can be established that the turn-on time of phase-*A* upper switch (T_{A-ON}) (see Fig. 18) can be expressed in the general form

$$T_{A-ON} = \frac{T_s}{4} + f(V^*)g(\alpha^*) \tag{20}$$

where T_s = sampling time (or period of switching frequency), V^* = command voltage magnitude, $f(V^*)$ = voltage amplitude function scale factor, α^* = angle in a sector (one of the six sectors), and $g(\alpha^*)$ = turn-on pulsewidth function at unit voltage amplitude. Fig. 18 shows the plot of (20) in all the six sectors with variable V^* covering UM and OM regions. In UM region, $f(V^*) = V^*$, and therefore, T_{A-ON} increases linearly with V^*. The UM region ends at the upper value of V^* when the maximum and minimum values of T_{A-ON} are $T_s/2$ and 0, respectively. At OM mode-1 and mode-2, T_{A-ON} saturates at the upper and lower levels as shown when it (shown by dotted curves) tends to exceed the limit values. Then, $f(V^*)$ becomes nonlinear function of V^*. The curves for T_{B-ON} and T_{C-ON} for the corresponding phases *B* and *C* are identical but mutually phase-shifted by 120°. The relation between $f(V^*)$ and V^* covering both UM and OM regions for dc link voltage $V_d^* = 300$ V is illustrated in Fig. 19. The linear characteristics in UM region $[f(\mathrm{V}^*) = V^*]$ ends at $V^* = 173$ V as shown. In OM region, the problem is to generate an appropriate relation between $f(V^*)$ and V^* so that linear transfer relation is maintained between V^* and actual inverter output. It can be shown that UM curve in Fig. 19 can be extended to OM mode-1 and mode-2 by using a nonlinear scale factor $f(V^*)(f(V^*) \neq V^*)$ that increases $f(V^*)$ nonlinearly in Fig. 19 until square-wave is reached at $V^* = \infty$ at square wave (i.e., $m' = 1$). The idea is the same as SPWM OM principle where modulation voltage is magnified. The data generated from Figs. 18 and 19 are used to train a backpropagation NNW shown in Fig. 20. Basically, it consists of two subnets. The angle subnet (lower part) receives the command voltage vector angle θ_e^* at the input and solves the turn-on pulse width functions $g_A(\alpha^*)$, $g_B(\alpha^*)$, and $g_C(\alpha^*)$ as shown. The upper voltage subnet receives the command voltage magnitude V^* and converts it to $f(V^*)$. Both the voltage subnet (1–3–1) and angle subnet (1–17–3) have three layers with sigmoidal TF in the hidden and output layers. The angle subnet outputs are multiplied by $f(V^*)$ and added with the bias signal $WT_s/4$ to generate the digital words corresponding turn-on time of the three phases. Then, a single UP/DOWN counter converts these words into the respective symmetrical pulse widths, as shown in the figure.

2) Three-Level Inverter: The NNW-based SVM principle for two-level inverter can be extended to a three-level inverter [23]–[25]. These references also discuss SVM technique of three-level inverter in UM and OM regions and their DSP-based implementation. The SVM of a three-level inverter is more complex than a two-level inverter because it is characterized by $3^3 = 27$ switching states. Each phase voltage output can be positive (*P*-state), negative (*N*-state), or zero (0) (neutral-point clamped) depending on the state of the phase-leg switches. The calculated turn-on times for *P*-state and *N*-state of a phase leg (U phase) for symmetrical pulse widths within a sample time (T_s) [25] are plotted in Fig. 21 for all the six sectors (*A*-*F*) with variable command voltage (V^*) and its angular position (θ_e), where T_{UP-ON} = turn-on time of P-state for phase U (or A) and T_{UN-ON} = turn-on time of N-state for phase-U. The curves for phases *V* and *W* are identical but are mutually phase-shifted by 120° The particular case considers dc link voltage $V_d = 3000$ V and $T_s = 1.0$ ms (i.e., 1.0 kHz switching frequency). Both UM and OM regions of the inverter are included in the figure. In UM mode-1, the curves remain unsaturated, but reaches saturation at 0.5 ms ($T_s/2$) at the end of this mode, and then mode-2 starts. In OM modes 1 and 2, part of the curves reach clamping level 0 as indicated. With increase in modulation factor, the curves become trapezoidal in shape and ultimately approach square-wave. The mapping of turn-on times in Fig. 21 as function of V^* and θ_e generate data for training the NNW. Fig. 22 shows the trained NNW topology for SVM implementation. The five-layer NNW uses 35 neurons. Both the input and output layers use bipolar linear TF, whereas the hidden layers use hyperbolic tan TF. The study indicated that a three-layer or four-layer NNW could also be

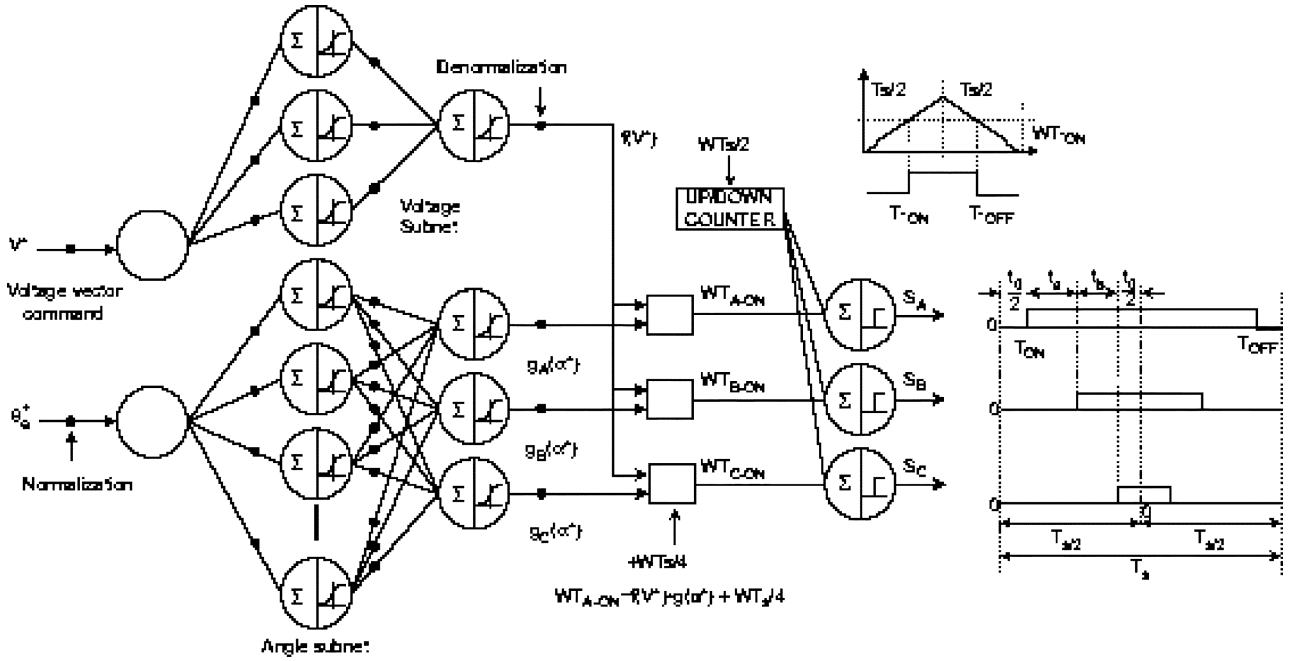

Fig. 20. Neural network topology for SVM of two-level inverter.

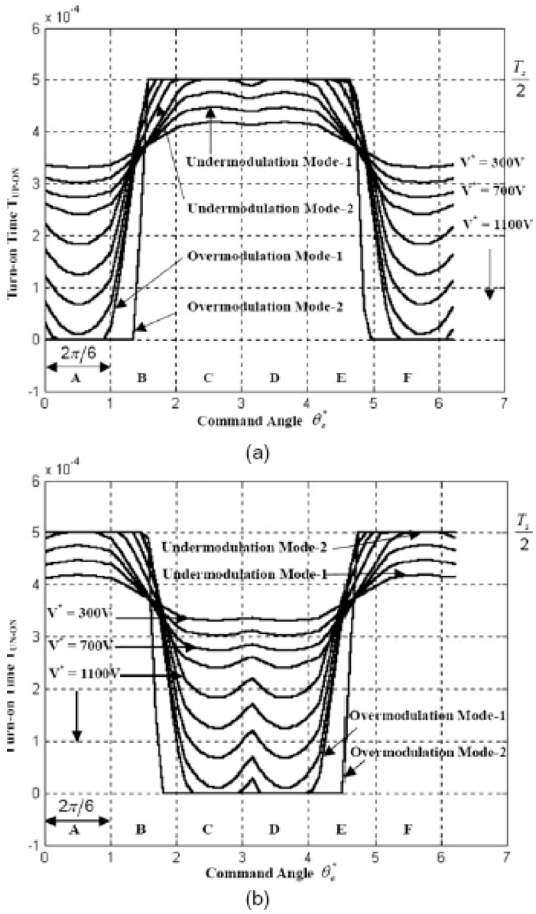

Fig. 21. Turn-on time plots for phase-U covering UM and OM regions. (a) T_{UP-ON} for P-state. (b) T_{UN-ON} for N-state.

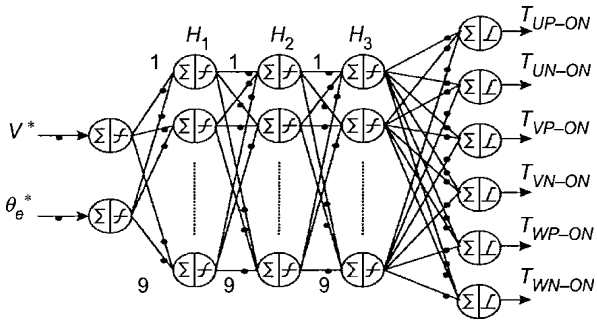

Fig. 22. Neural network topology for SVM of three-level inverter.

used, but these required larger number of neurons. The NNW in Fig. 22 generates the digital words corresponding to turn-on times of the phases which are converted to symmetrical pulse width signals through an interface logic circuit and a central UP/DOWN counter, as shown in Fig. 23 [25]. The detailed processing of the T_{UP-ON} signal, for example, is shown in the upper part of the figure. The signal is segmented sectorwise with a sector identification signal (not shown) such that T_{UP1} component is clamped to zero in even sectors (*B*, *D*, and *F*), whereas T_{UP2} component is clamped to zero in odd sectors (*A*, *C*, and *E*). This segmentation complexity arises because it can be shown analytically that in the odd sectors, the *P* states appear as pulsed waves and the *N* states appear as notched waves, whereas in even sectors, their roles are reversed. The signal T_{UP1} is compared with the output of the UP/DOWN counter to generate the logic pulse *A*, whereas T_{UP2} is compared with the inverted output of the counter to generate the logic pulse *B*. These pulses are then logically ANDed to generate the pulse width for *P* state of *U* phase. Similar segmentation and processing are done for all the six output channels of the NNW.

3) Five-Level Inverter: In the final phase of discussion, NNW-based SVM implementation will be discussed for a five-level inverter [26] which is exceedingly complex. Fig. 24 shows the simplified representation of diode-clamped five-level inverter, where the synthesis of output phase voltages from

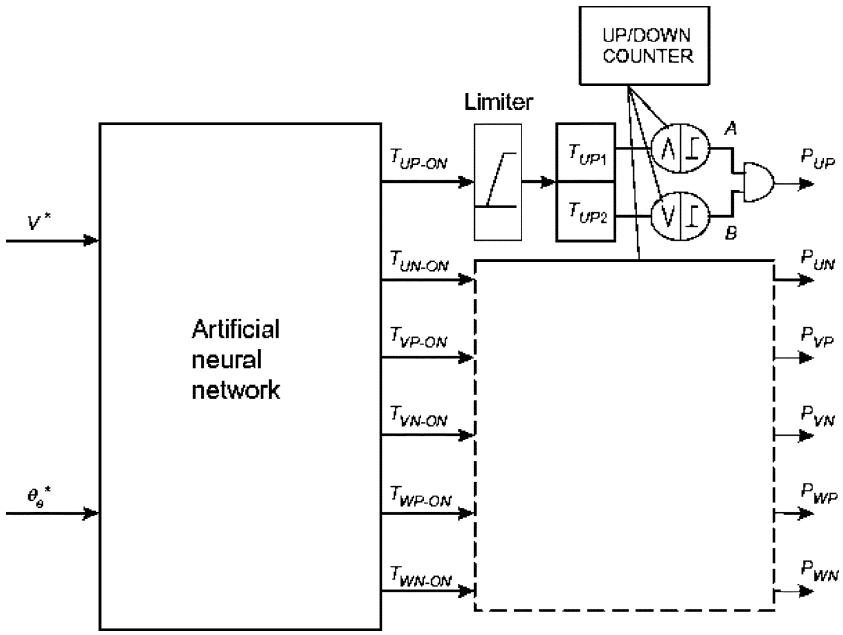

Fig. 23. Neural network with simplified interface logic and counter to generate pulse width signals.

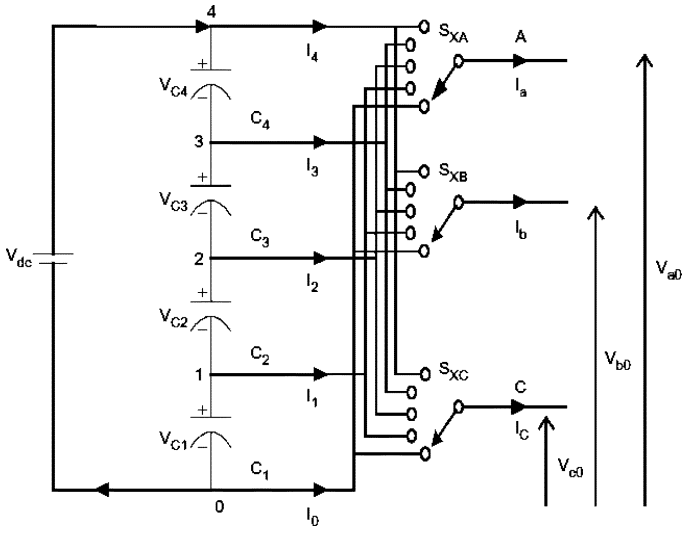

Fig. 24. Simplified representation of five-level inverter.

the five levels of capacitor voltages is indicated through five-position multiplexer switches. The hexagon switching states of the inverter are indicated in Fig. 25. The inverter has $5^3 = 125$ switching states, 96 triangles, 61 effective space vectors, and multiplicity of switching states for inner triangle corners as shown. The six sectors of the hexagon are numbered *A* to *F*, respectively. The sector *A* is drawn separately in Fig. 25(b) showing its 16 triangles and the switching intervals (t_a, t_b, and t_c) for the corner vectors, where $t_a + t_b + t_c = T_s$ (sampling time). The command voltage trajectory for modulation factor $m' = 0.53$ is shown on the hexagon. The location of the command vector in a triangle (T_{hex}) of the hexagon can be determined by the equation

$$T_{\text{hex}} = (\text{Sector No.} - 1) \cdot 16 + T_r \tag{21}$$

Fig. 25. (a) Switching states of five-level inverter. (b) Sector-A triangle showing switching states.

where $T_r = \text{number of the corresponding triangle}$ located in sector *A*. In fact, pulse width information of SVM can be obtained by identifying the command vector location in any of the 96 triangles from the principles of input-output nonlinear mapping. Fig. 26 shows the typical levels of the output voltage (Levels 0 to 4) for phase-*A* and the corresponding synthesis of pulse widths in the four levels are given by

$$T_{4A} = K_{a4A}(T_{\text{hex}}) \cdot t_a + K_{b4A}(T_{\text{hex}}) \cdot t_b + K_{c4A}(T_{\text{hex}}) \cdot t_c \tag{22}$$

$$T_{3A} = K_{a3A}(T_{\text{hex}}) \cdot t_a + K_{b3A}(T_{\text{hex}}) \cdot t_b + K_{c3A}(T_{\text{hex}}) \cdot t_c \tag{23}$$

$$T_{2A} = K_{a2A}(T_{\text{hex}}) \cdot t_a + K_{b2A}(T_{\text{hex}}) \cdot t_b + K_{c2A}(T_{\text{hex}}) \cdot t_c \tag{24}$$

$$T_{1A} = K_{a1A}(T_{\text{hex}}) \cdot t_a + K_{b1A}(T_{\text{hex}}) \cdot t_b + K_{c1A}(T_{\text{hex}}) \cdot t_c \tag{25}$$

where t_a, t_b, and $t_c = \text{time intervals}$ of the corner switching states for the T_{hex} location of the vector, and K_{a4A}, K_{b4A}, etc. = coefficients (function of T_{hex}) of time intervals that determine the pulse widths. These coefficients (12 for each phase) for the three phases can be generated by prior computation and storing. Fig. 27 shows the NNW-based SVM implementation for the complete inverter in the UM range only. The NNW is segmented into two parts: the first one is the Triangle Identification NNW (2–3–3–1) and the second one is the Duty Cycle Calculation NNW (2–10–2) as shown. The first NNW receives the command voltage magnitude (V^*) and

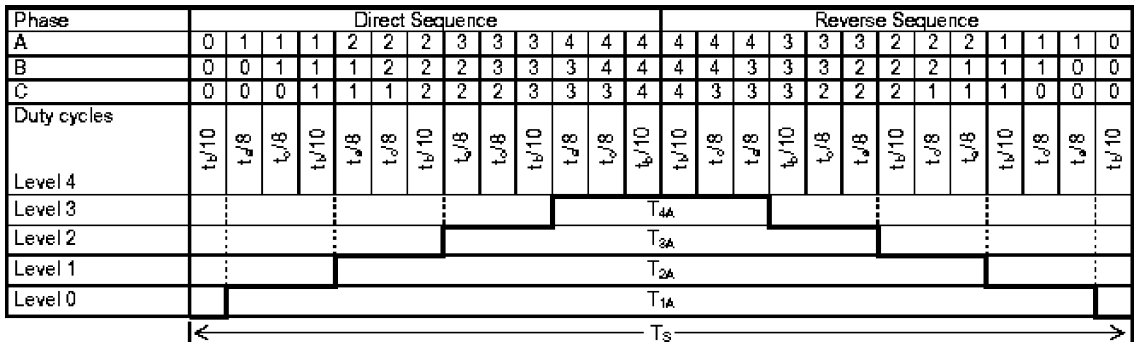

Fig. 26. Pulse width synthesis for phase-A voltage in a sampling interval T_s. [For triangle 1 of Fig. 25(a)Fig.25(a)].

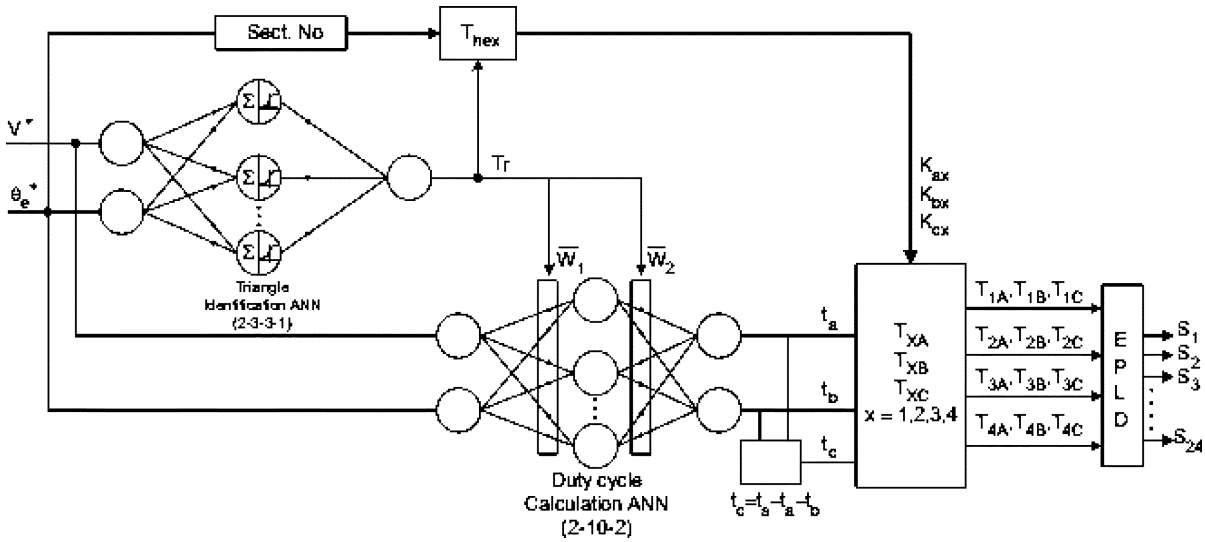

Fig. 27. Block diagram of neural network-based SVM of five-level inverter.

the vector angle (θ_e^*) and identifies the vector location in one of the equivalent triangles (T_r) in sector A. This information is then used to calculate $T_{\rm hex}$ by solving (21) with the sector location as indicated. Each of the 16 numbers for T_r also generates a pair of weight vectors $\bar{W}_1$ and $\bar{W}_2$ by a look-up table and loads the second NNW as shown. The Duty Cycle NNW estimates the time intervals t_a, t_b, and t_c for location of the command voltage vector in the hexagon.. Again, each of the 96 $T_{\rm hex}$ locations stores the coefficients as indicated. These coefficients are used to calculate the three phase pulse widths as discussed before. Finally, the EPLD translates these digital words into the corresponding switching signals of 24 devices ($S_1 - S_{24}$). For selecting the switching pulses, the inverter capacitor voltage balancing and pulsing/notching of odd/even sectors are considered [25] but are not discussed here. Each component NNW was trained from the simulation data and then validated by test. The complexity of NNW-based SVM implementation in this case is apparent. However, it is simpler and faster than DSP-based implementation.

E. Speed Estimation

The speed estimation by MRAC technique has been discussed in the literature [28]. In this method, the voltage model flux estimation (defined as reference model) from the machine stator equations is compared with the speed adaptive current model (machine rotor equations) to tune the motor speed. The current model equations can be discretized and expressed in the general form [6]

$$\begin{bmatrix} \psi_{dr}^s(k+1) \\ \psi_{qr}^s(k+1) \end{bmatrix} = \begin{bmatrix} W_{11} & W_{21} \\ W_{12} & W_{22} \end{bmatrix} \begin{bmatrix} \psi_{dr}^s(k) \\ \psi_{qr}^s(k) \end{bmatrix} + \begin{bmatrix} W_{31} & 0 \\ 0 & W_{32} \end{bmatrix} \begin{bmatrix} i_{ds}^s(k) \\ i_{qs}^s(k) \end{bmatrix} \quad (26)$$

where $W_{11} = 1 - T_s/T_r$, $W_{21} = -\omega_r T_s$, $W_{12} = \omega_r T_s$, $W_{22} = 1 - T_s/T_r$, $W_{31} = W_{32} = L_m \cdot T_s/T_r$, $T_s =$ sampling time, $T_r =$ rotor time constant, $\omega_r =$ speed, and $L_m =$ magnetizing inductance. Equation (26) can be represented by a real-time RNN shown in Fig. 28 [29] using bipolar linear TFs for the neurons. The solution of the voltage model generates the desired flux components as shown. These signals are compared with the RNN output signals and the weights are trained online so that the error $\xi(k+1)$ tends to be zero. It is assumed that the voltage model and current model parameters remain invariant. Note that among the six weights of the RNN, only W_{21} and W_{12} (circled) are function of ω_r, and therefore, these weights only are to be trained. However, if all the weights are trained, the speed as well as rotor time constant (T_r) can be tuned [29] .

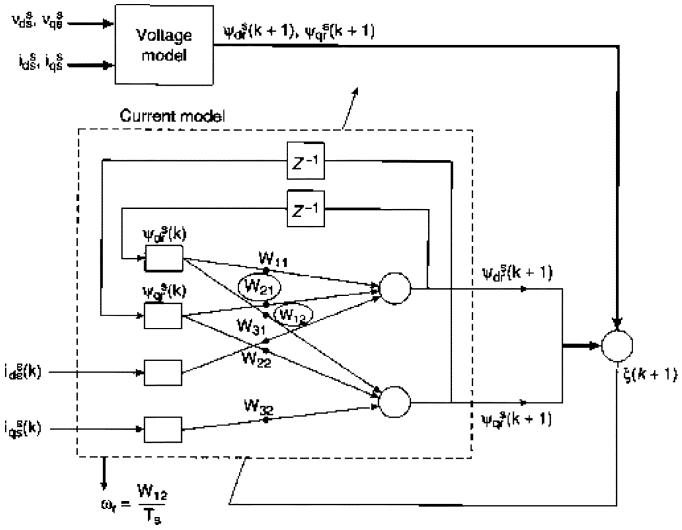

Fig. 28. Real-time recurrent network-based speed estimation.

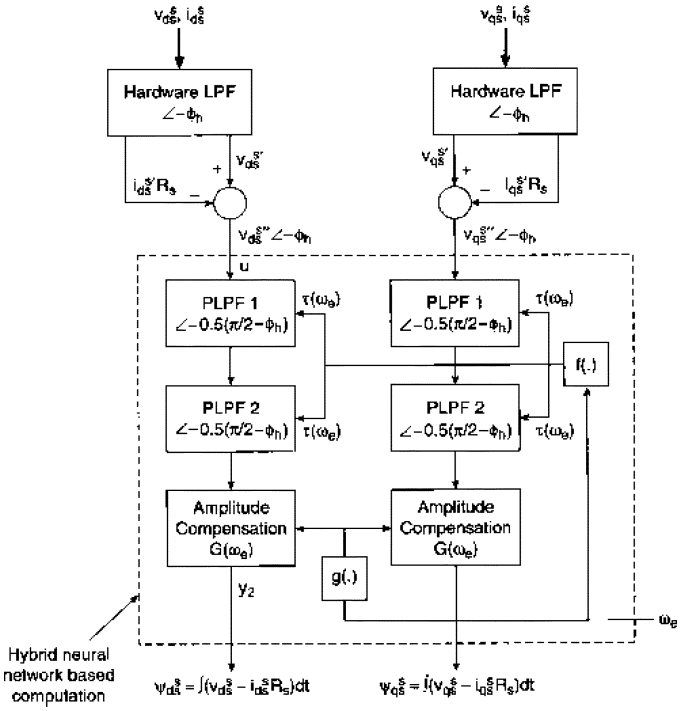

Fig. 29. Two-stage programmable cascaded low-pass filter (PCLPF) for stator flux vector estimation.

F. Adaptive Flux Vector Estimation

Adaptive stator flux vector estimation in wide frequency range by multistage programmable cascaded low-pass filter (PCLPF) has been discussed in the literature [31]. The PCLPF basically uses the voltage model for flux estimation but avoids the ideal integration, where offset buildup at the output is a problem at low frequency. Fig. 29 shows a simplified two-stage PCLPF where both the channels for computation of ψ_{ds}^s and ψ_{qs}^s are identical. The machine terminal voltages and currents are sensed, converted to $d^s - q^s$ frame, filtered from harmonic ripple, and stator resistance drop is subtracted from each channel by using op-amps, as indicated in the figure. The resulting voltages have lagging phase-shift angle $-\varphi_h$ that increases with frequency. Each PLPF stage shown in the dotted box is basically a first-order low-pass filter (LPF) in the form $(K/(1+\tau S)$, where the time constant τ and gain K are function of frequency (ω_e). For the total $-90°$ phase-shift angle, each PLPF stage is designed with phase-shift angle $\angle -0.5(\pi/2-\varphi_h)$, and the desired magnitude attenuation remain invariant with frequency. Therefore, the amplitude compensation G and τ are programmed with frequency. Ideally, if $\varphi_h = 0$, each PLPF is required to give 45° phase shift. Splitting the amplitude compensation $G(\omega_e)$ and merging $\sqrt{G}$ with identical PLPF stages in each channel, the resulting discretized equations can be written in the general form

$$\begin{bmatrix} y_1(k+1) \\ y_2(k+1) \end{bmatrix} = \begin{bmatrix} W_{11} & 0 \\ W_{21} & W_{22} \end{bmatrix} \begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} + \begin{bmatrix} W_{13} \\ 0 \end{bmatrix} u(k) \tag{27}$$

where $W_{11} = W_{22} = 1 - T_s/\tau$, $W_{21} = W_{13} = \sqrt{G}T_s/\tau$ and $T_s = \text{sampling time}$. The (27) for each channel in the dotted enclosure can be implemented by a real-time RNN (see Fig. 5) as shown in the upper part of Fig. 30(a) [32]. The RNN neurons have bipolar linear TFs and all weights are trainable as function of frequency ω_e. For each sinusoidal input voltage wave at a certain frequency, there is a corresponding target output flux wave at $(90° - \varphi_h)$ lag angle and the RNN weights are trained to satisfy this relation. Therefore, the RNN is adaptive, i.e., its weights vary with frequency. The training data was generated by simulation in the frequency range of 0.01–200 Hz with the step size of 0.5 Hz. The trained weights as function of frequency (slightly nonlinear) are shown in Fig. 30(b). An additional backpropagation NNW shown in the lower part generates these weights as function of ω_e. This NNW uses linear TF at the output layer but sigmoidal TF in the hidden layer. Note that ω_e for the hybrid NNW is estimated (see (36)) in circulatory manner by using the output flux vector signals. The principle discussed here can easily be applied for rotor flux vector estimation.

G. Induction Motor Vector Drive With Neural Network-Based SVM and Flux Estimation

The backpropagation NNW-based SVM and RNN-based stator flux vector estimation described in Figs. 20 and 29, respectively, are integrated in a core EV drive shown in Fig. 31 [31], [33]. As usual, the torque control is considered in the outer loop with close loop stator flux control. The flux loop requires injection of decoupling current i_{dq} as shown. The synchronous current control loops generate the voltage commands to the vector rotator which correspondingly generates the voltage vector for the SVM modulator. The PCLPF integrator, as described before, generates the stator flux vector components from the machine terminal voltages and currents. The signal estimation block solves the following equations by a backpropagation NNW (not shown) using the same principle (see Fig. 17) discussed before

$$\hat{\psi}_s = \sqrt{(\psi_{ds}^s)^2 + (\psi_{qs}^s)^2} \tag{28}$$

$$\theta_e = \sin^{-1}\left(\frac{\psi_{qs}^s}{\hat{\psi}_s}\right) \tag{29}$$

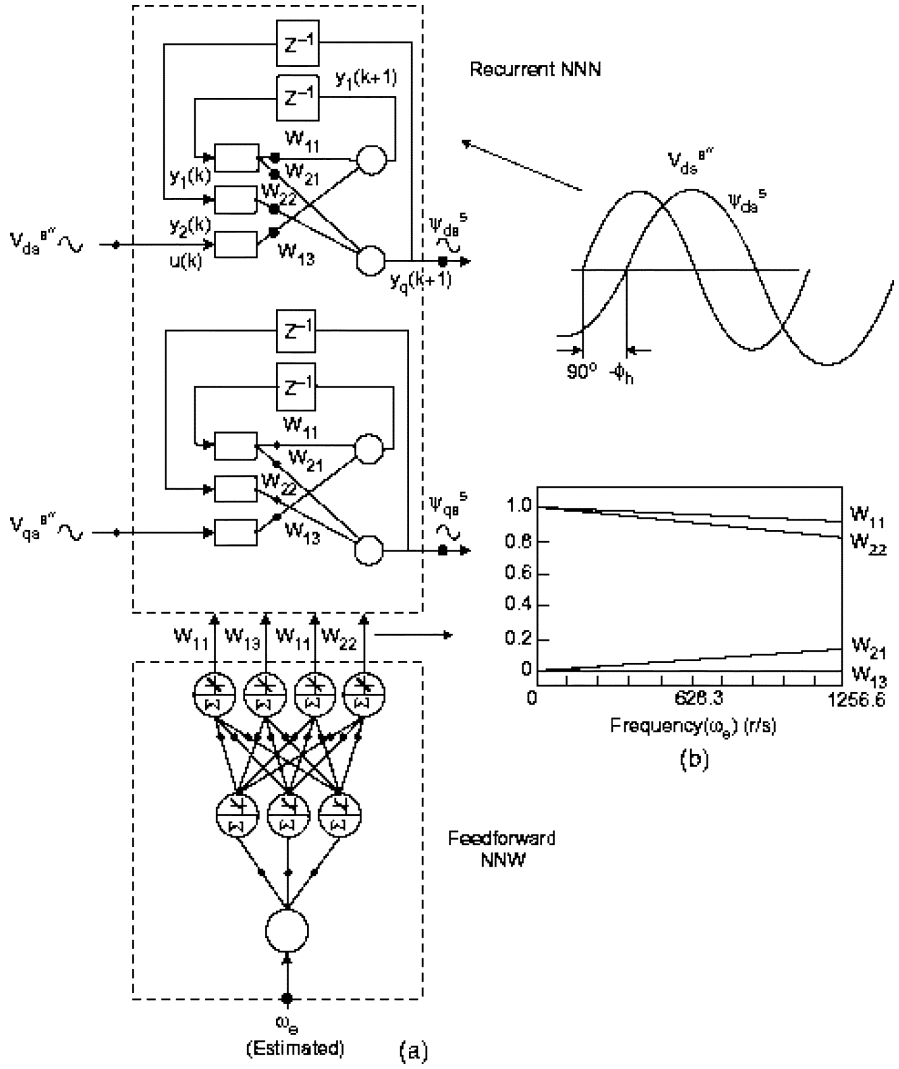

Fig. 30. (a) Hybrid neural network for stator flux vector estimation. (b) RNN weights as function of frequency.

$$\psi_{qs} = \psi_{qs}^{s} \cos\theta_e - \psi_{ds}^{s} \sin\theta_e \tag{30}$$

$$\psi_{ds} = \psi_{qs}^{s} \sin\theta_e + \psi_{ds}^{s} \cos\theta_e \tag{31}$$

$$i_{qs} = i_{qs}^{s} \cos\theta_e - i_{ds}^{s} \sin\theta_e \tag{32}$$

$$i_{ds} = i_{qs}^{s} \sin\theta_e + i_{ds}^{s} \cos\theta_e \tag{33}$$

$$i_{dq} = \frac{\sigma L_s i_{qs}^2}{\psi_{ds} - \sigma L_s i_{ds}} \tag{34}$$

$$T_e = \frac{3P}{4} \left[\psi_{ds} i_{qs} - \psi_{qs} i_{ds}\right] \tag{35}$$

$$\omega_e = \frac{\left[(v_{qs}^{s} - i_{qs}^{s} R_s)\psi_{ds}^{s} - (v_{ds}^{s} - i_{ds}^{s} R_s)\psi_{qs}^{s}\right]}{\hat{\psi}_s^2} \tag{36}$$

where all standard symbols are used. The drive operates satisfactorily in the constant torque region starting from very low frequency (0.01 Hz).

VI. Conclusion

This paper gives a broad and comprehensive discussion on neural network principles and their applications in power electronics and motor drives, that includes several application examples which are mainly based on the author's own contribution. Neural network applications in this area have surged significantly in the recent literature. However, the paper is not intended to be a state-of-the-art technology review for these applications. Many additional references are included (but not cited in the text) in this paper which can be digged further for information on additional applications. Neural network itself is a vast discipline in artificial intelligence, and the basic technology has advanced tremendously in recent years. Considering the immense potentiality of neural networks in future, their applications in power electronics area are yet in the stage of infancy. The paper focuses the discussion mainly on the feedforward backpropagation network and its applications, since it is the most commonly used topology in power electronics. Of course, real-time recurrent network and its applications have also been covered. There are many other feedforward and recurrent network topologies which require systematic exploration for their applications in power electronics. Besides, powerful intelligent control and estimation techniques can be developed using hybrid AI (neuro-fuzzy, neuro-genetic, and neuro-fuzzy-genetic) systems. In spite of the technology advancement, currently, in-

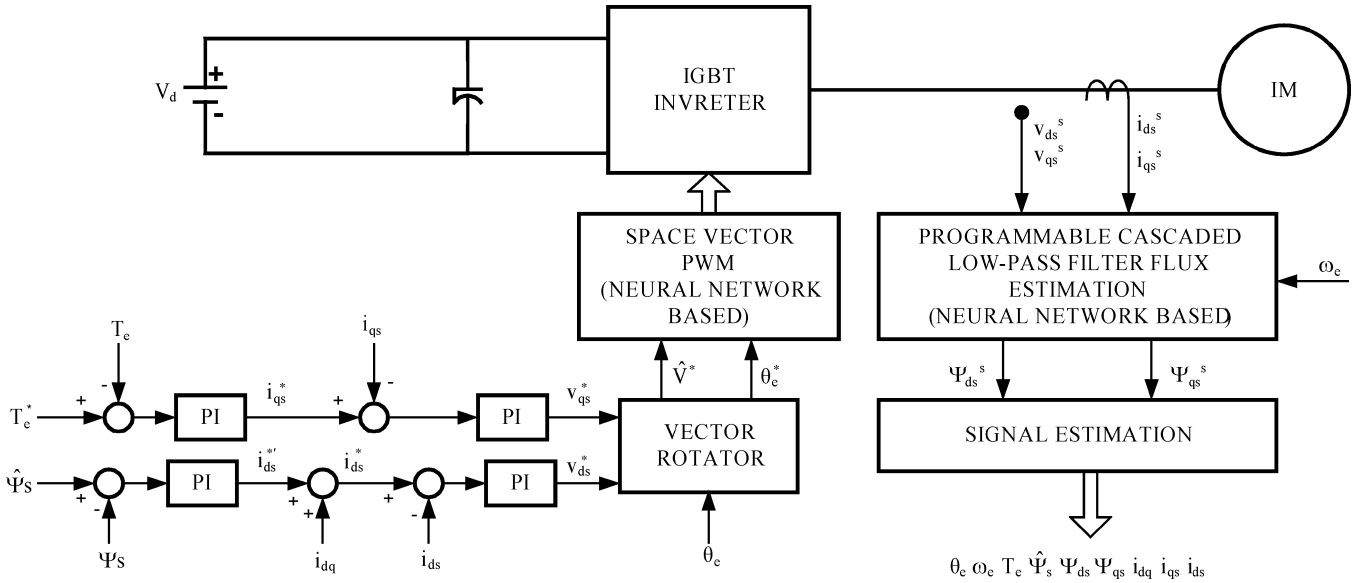

Fig. 31. Stator flux oriented induction motor vector drive with neural network-based SVM and flux estimation.

dustrial applications of neural networks in power electronics appear to be very few. At present, most neural network implementations are based on slow and sequentially executed DSPs. Multiple DSPs have been used to enhance the execution speed. Of course, the speed of DSP is improving dramatically in recent years and its cost is falling sharply. Concurrently, there has been tremendous advancement of VLSI technology, and the ASIC chips including FPGAs have been applied to a limited extent for neural network implementation. However, large and economical digital ASIC chips, particularly designed for neural networks, are not yet available. This is possibly impeding the widespread industrial applications of neural network. With the current trend of R&D, it is expected that viable and inexpensive neural network ASIC chips will be available in future, and this will promote widespread neural network applications in power electronics and motor drives.

ACKNOWLEDGMENT

The author would like to thank Dr. J. O. P. Pinto of Federal University of Mato do Sul(UFMS), Brazil and Dr. M. Kazmierkowski of Warsaw University of Technology, Poland for their help in preparation of this paper. Thanks are also due to my graduate students, research scholars, and visiting professors who did research and published papers while working in the Power Electronics Research Laboratory, University of Tennessee. This paper is mainly based on their work.

REFERENCES

[1] S. Haykin, *Neural Networks*. New York: Macmillan, 1994.
[2] *MathWorks Neural Network Toolbox User's Guide,* 2001.
[3] B. K. Bose, *Power Electronics and Motor Drive – Advances and Trends*. Burlington, MA: Elsevier, 2006.
[4] M. N. Cirstea, A. Dinu, J. G. Khor, and M. McCormick, *Neural and Fuzzy Logic Control of Drives and Power Systems*. Burlington, MA: Elsevier, 2002.
[5] H. S. Tsoukalas and R. E. Uhrig, *Fuzzy and Neural Approaches in Engineering*. New York: Wiley, 1997.
[6] B. K. Bose, *Modern Power Electronics and AC Drives*. Upper Saddle River, NJ: Prentice-Hall, 2002.
[7] ——, "Expert system, fuzzy logic, and neural network applications in power electronics and motion control," *Proc. IEEE*, vol. 82, pp. 1303–1323, Aug. 1994.
[8] B. Burton, F. Karman, R. G. Harley, T. G. Habetler, M. A. Brooke, and R. Poddar, "Identification and control of induction motor station currents using fast on line random training neural networks," *IEEE Trans. Ind. Appl.*, vol. 33, pp. 697–704, May/Jun. 1997.
[9] K. J. Hunt *et al.*, "Neural networks for control systems – A survey," *Automatica*, vol. 28, pp. 1083–1112, 1992.
[10] B. Burton, R. G. Harley, and T. G. Habetler, "The application and hardware implementation of continuously online trained feedforward neural networks for fast adaptive control and prediction in power electronic systems," in *Proc. Conf. Rec. IEEE FEPPCON III*, South Africa, Jul. 1998, pp. 215–224.
[11] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 4–27, Mar. 1990.
[12] P. Z. Grabowski, M. P. Kazmierkowski, B. K. Bose, and F. Blaabjerg, "A simple direct-torque neuro-fuzzy control of PWM-inverter-fed induction motor drive," *IEEE Trans. Ind. Electron.*, vol. 47, no. 4, pp. 863–870, Aug. 2000.
[13] B. K. Bose, "Artificial neural network applications in power electronics," in *Proc. Conf. Rec. IEEE IECON*, 2001, pp. 1631–1638.
[14] J. O. P. Pinto, "Artificial neural networks based three-phase sine function generation–Demo program," *Internal Memo*, Jan. 2005.
[15] B. K. Bose, "Artificial intelligence techniques–A new and advancing frontier in power electronics and motor drives," in *Proc. Int. Power. Electron. Conf.*, Niigata, Japan, Apr. 2005, pp. 100–109.
[16] B. Burton, R. G. Harley, G. Diana, and J. L. Rodgerson, "Implementation of a neural network to adaptively identify and control VSI-Fed induction motor stator currents," *IEEE Trans. Ind. Appl.*, vol. 34, no. 3, pp. 580–588, May/Jun. 1998.
[17] J. Zhao and B. K. Bose, "Neural network based waveform processing and delayless filtering in power electronics and ac drives," *IEEE Trans. Ind. Electron.*, vol. 51, no. 5, pp. 981–991, Oct. 2004.
[18] M. G. Simoes and B. K. Bose, "Neural network based estimation of feedback signals for a vector controlled induction motor drive," *IEEE Trans. Ind. Appl.*, vol. 31, no. 3, pp. 620–629, May/Jun. 1995.
[19] F. Harashima, Y. Demizu, S. Kondo, and H. Hoshimoto, "Application of neural networks to power converter control," in *Proc. IEEE Ind. Appl. Soc. Annu. Meeting*, 1989, pp. 1087–1091.
[20] M. Buhl and R. D. Lorenz, "Design and implementation of neural networks for digital current regulation of inverter drives," in *Proc. IEEE Ind. Appl. Soc. Annu. Meeting*, 1991, pp. 415–421.
[21] M. P. Kazmierkowski and D. Sobczuk, "Improved neural network current regulator for VC-PWM inverters," in *Proc. Conf. Rec. IEEE IECON*, 1994, pp. 1237–1241.

[22] J. O. P. Pinto, B. K. Bose, L. E. B. daSilva, and M. P. Kazmierkowski, "A neural network based space vector PWM controller for voltage-fed inverter induction motor drive," *IEEE Trans. Ind. Appl.*, vol. 36, no. 6, pp. 1628–1636, Nov./Dec. 2000.

[23] S. Mondal, J. O. P. Pinto, and B. K. Bose, "A neural network based space vector PWM controller for a three-level voltage-fed inverter induction motor drive," *IEEE Trans. Ind. Appl.*, vol. 38, no. 3, pp. 660–669, May/Jun. 2002.

[24] S. K. Mondal, B. K. Bose, V. Oleschuk, and J. O. P. Pinto, "Space vector pulse width modulation of three-level inverter extending operation into overmodulation region," *IEEE Trans. Power Electron.*, vol. 18, no. 2, pp. 604–611, Mar. 2003.

[25] C. Wang, B. K. Bose, V. Oleschuk, S. Mondal, and J. O. P. Pinto, "Neural network based SVM of a three-level inverter covering overmodulation region and performance evaluation on induction motor drives," in *Proc. Conf. Rec. IEEE IECON*, 2003, pp. 1–6.

[26] N. P. Filho, J. O. P. Pinto, B. K. Bose, and L. E. B. daSilva, "A neural network based space vector PWM of a five-level voltage-fed inverter," in *Proc. Conf. Rec. IEEE Ind. Appl. Soc. Annu. Meeting*, 2004, pp. 2181–2187.

[27] D. Daniolos, M. K. Darwish, and P. Mehta, "Optimized PWM inverter control using artificial neural networks," *Electron. Lett.*, vol. 31, no. 20, pp. 1739–1740, Sep. 1995.

[28] C. Schauder, "Adaptive speed identification for vector control of induction motors without rotational transducers," *IEEE Trans. Ind. Appl.*, vol. 28, no. 5, pp. 1054–1061, Sep./Oct. 1992.

[29] B. Jayanand, "Simulation studies on speed sensorless operation of vector controlled induction motor drives using neural network," Ph.D. dissertation, IIT, Madras, India, 1997.

[30] L. Ben-Brahim and R. Kurosawa, "Identification of induction motor speed using neural networks," in *Proc. IEEE PCC*, Yokohama, 1993, pp. 689–694.

[31] B. K. Bose and N. R. Patel, "A sensorless stator flux oriented vector controlled induction motor drive with neuro-fuzzy based performance enhancement," in *Proc. IEEE Ind. Appl. Soc. Annu. Meeting*, 1997, pp. 393–400.

[32] L. E. B. daSilva, B. K. Bose, and J. O. P. Pinto, "Recurrent neural network based implementation of a programmable cascaded low pass filter used in stator flux synthesis of vector controlled induction motor drive," *IEEE Trans. Ind. Electron.*, vol. 46, no. 3, pp. 662–665, Jun. 1999.

[33] J. O. P. Pinto, B. K. Bose, and L. E. Borges, "A stator flux oriented vector controlled induction motor drive with space vector PWM and flux vector synthesis by neural networks," *IEEE Trans. Ind. Appl.*, vol. 37, no. 5, pp. 1308–1318, Sep./Oct. 2001.

[34] M. H. Kim, M. G. Simoes, and B. K. Bose, "Neural network based estimation of power electronic waveforms," *IEEE Trans. Power Electron.*, vol. 11, no. 2, pp. 383–369, Mar. 1996.

[35] B. Karanayil, M. F. Rahman, and C. Grantham, "Rotor resistance identification using artificial neural networks for an indirect vector controlled induction motor drive," in *Proc. Conf. Rec. IEEE IECON*, vol. 1001, pp. 1315–1320.

[36] ——, "Stator and rotor resistance observers for induction motor driven using fuzzy logic and neural networks," *IEEE Trans. Energy Conv.*, vol. 20, no. 4, pp. 771–780, Dec. 2005.

[37] B. Singh, V. Verma, and J. Solanki, "Active power filter based selective compensation of current using neural network," in *Proc. Conf. Rec. IEEE ISIE*, 2004, pp. 104–110.

[38] S. M. R. Rafiei, R. Ghazi, and H. A. Taliyat, "IEEE-519-based real-time and optimal control of active filters under nonsinusoidal line voltages using neural networks," *IEEE Trans. Power Del.*, vol. 17, no. 3, pp. 815–821, Jul. 2002.

[39] F. Temurtas, R. Gunturkun, N. Yumusaka, and H. Temurtas, "Harmonic detection using feedforward and recurrent neural networks for active filters," *Electric Power Syst. Res.*, vol. 72, pp. 33–40, 2004.

[40] Y. Wang, J. Gu, and C. Chen, "An improved adaline algorithm for online tracking of harmonic components," *Int. J. Power Energy Syst.*, vol. 23, pp. 117–127, 2003.

[41] D. O. Abdeslam, P. Wira, J. Merckle, and Y. A. Chapuis, "A neural approach for the control of an active power filter," in *Proc. Conf. Rec. IPEC*, Niigata, Japan, 2005, pp. 923–929.

[42] T. Hiyama *et al.*, "Evaluation of neural network based real time maximum power tracking controller in PV system," *IEEE Trans. Energy Conv.*, vol. 10, no. 3, Sep. 1995.

[43] M. A. El-Sharkawi, "High performance brushless motors using neural network," in *Proc. IEEE PES Summer Conf.*, Jul. 1993, pp. 200–207.

[44] S. Wersooriya and M. A. El-Sharkawi, "Identification and control of a dc motor using backpropagation neural network," *IEEE Trans. Energy Convers.*, vol. 6, pp. 663–669, Dec. 1991.

[45] Y. S. Kung, C. M. Liaw, and M. S. Ouyang, "Adaptive speed control for induction motor drives using neural networks," *IEEE Trans. Ind. Electron.*, vol. 42, no. 1, pp. 25–32, Feb. 1995.

[46] K. L. Shi, T. F. Chan, Y. K. Wong, and S. L. Ho, "Direct self control of induction motor based on neural network," *IEEE Trans. Ind. Appl.*, vol. 37, no. 5, pp. 1290–1298, Sep.-Oct. 2001.

[47] A. Rabaai, R. Kotaru, and M. D. Kankam, "Online training of parallel neural network estimators for control of induction motors," *IEEE Trans. Ind. Appl.*, vol. 37, no. 5, pp. 1512–1521, Sep.-Oct. 2001.

[48] T. C. Chen and T. T. Sheu, "Model referencing neural network controller for induction motor speed control," *IEEE Trans. Energy Conv.*, vol. 17, no. 2, pp. 157–163, Jun. 2002.

[49] M. Wlas, Z. Krzeminski, J. Guzinski, H. Abu-Rub, and H. A. Tolyat, "Artificial-neural-network-based sensorless nonlinear control of induction motors," *IEEE Trans. Energy Conv.*, vol. 20, no. 3, pp. 520–528, Sep. 2005.

[50] S. H. Kim, T. S. Park, J. Y. Yoo, and G. T. Park, "Speed sensorless vector control of an induction motor using neural network speed estimation," *IEEE Trans. Ind. Electron.*, vol. 48, no. 3, pp. 609–614, Jun. 2001.

[51] M. Mohamadian, E. Nowicki, F. Ashrafzadeh, A. Chu, R. Sachdeva, and E. Evanik, "A novel neural network controller and its efficient DSP implementation for vector controlled induction motor drives," *IEEE Trans. Ind. Appl.*, vol. 39, no. 6, pp. 1622–1629, Nov./Dec. 2003.

[52] C. M. Kwan and F. L. Lewis, "Robust backstepping control of induction motors using neural networks," *IEEE Trans. Neural Networks*, vol. 11, no. 5, pp. 1178–1187, Sep. 2000.

[53] F. J. Lin, R. J. Wai, W. D. Chou, and S. P. Hsu, "Adaptive backstepping control using recurrent neural network for linear induction motor drive," *IEEE Trans. Ind. Electron.*, vol. 49, no. 1, pp. 134–146, Feb. 2002.

[54] A. Rubaai and R. Kotaru, "Online identification and control of a DC motor using learning adaptation of neural networks," *IEEE Trans. Ind. Appl.*, vol. 33, no. 3, pp. 935–942, May-Jun. 2000.

[55] E. N. Sanchez, A. G. Loukianov, and R. A. Felix, "Dynamic triangular neural controller for stepper motor trajectory tracking," *IEEE Trans. Syst. Man, Cybern.*, vol. 32, no. 1, pt. C, pp. 24–30, Feb. 2002.

[56] A. Rabaai and R. Kotaru, "Adaptation learning control scheme for a high performance permanent-magnet stepper motor using online random training of neural networks," *IEEE Trans. Ind. Appl.*, vol. 37, no. 2, pp. 495–502, Mar.-Apr. 2001.

[57] G. J. Wang, C. T. Fong, and K. J. Chang, "Neural network based self-tuning PI controller for precise motion control of PMAC motors," *IEEE Trans. Ind. Electron.*, vol. 48, no. 2, pp. 408–415, Apr. 2001.

[58] H. J. Guo, S. Sagawa, T. Watanabe, and O. Ichinokura, "Sensorless driving method of permanent-magnet synchronous motors based on neural networks," *IEEE Trans. Magn.*, vol. 39, no. 5, pt. 2, pp. 3247–3249, Sep. 2003.

[59] F. J. Lin, R. J. Wai, and H. P. Chen, "A PM synchronous motor drive with an on-line trained fuzzy neural network controller," *IEEE Trans. Energy Conv.*, vol. 13, pp. 319–325, 1998.

[60] T. D. Batzel and K. Y. Lee, "An approach to sensorless operation of the permanent–Magnet synchronous motor using diagonally recurrent neural networks," *IEEE Trans. Energy Conv.*, vol. 18, pp. 100–106, Mar. 2003.

[61] Y. Yang, D. M. Vilathgamuwa, and M. A. Rahman, "Implementation of an artificial-neural-network-based real-time adaptive controller for an interior permanent-magnet motor drive," *IEEE Trans. Ind. Appl.*, vol. 39, no. 1, pp. 96–104, Jan./Feb. 2003.

[62] T. Pajchrowski, K. Urbanski, and K. Zawirski, "Robust speed control of PMSM servodrive based on ANN application," in *Proc. 10th Eur. Conf. Power Electron. Appl.*, Toulouse, Sep. 2–4, 2003, 833.

[63] T. Pajchrowski and K. Zawirski, "Robust speed control of servodrive based on ANN," in *Proc. Conf. Rec. IEEE ISIE*, 2005, pp. 81–86.

[64] R. J. Wai, "Total sliding mode controller for PM synchronous servo motor drive using recurrent fuzzy neural network," *IEEE Trans. Ind. Electron.*, vol. 48, no. 5, pp. 926–944, Oct. 2001.

[65] A. Rubaai, D. Ricketts, and M. D. Kankam, "Development and implementation of an adaptive fuzzy- neural-network controller for brushless drives," *IEEE Trans. Ind. Appl.*, vol. 38, no. 2, pp. 441–447, Mar./Apr. 2002.

[66] L. Wenzhe, A. Keyhani, and A. Fardoun, "Neural network based modeling and parameter identification of switched reluctance motors," *IEEE Trans. Energy Conv.*, vol. 18, no. 2, pp. 284–290, Jun. 2003.
[67] E. Mese and D. A. Torry, "An approach for sensorless position estimation for switched reluctance motors using artificial neural networks," *IEEE Trans. Power Electron.*, vol. 17, pp. 66–75, Jan. 2002.
[68] X. Q. Liu, H. Y. Zhang, J. Liu, and J. Yang, "Fault detection and diagnosis of permanent magnet DC motor based on parameter estimation and neural network," *IEEE Trans. Ind. Electron.*, vol. 47, no. 5, pp. 1021–1030, Oct. 2000.
[69] R. M. Tallam, T. G. Habetler, and R. G. Harley, "Continual on-line training of neural networks with applications to electric machine fault diagnostics," in *Proc. Conf. Rec. IEEE PESC*, 2001, vol. 4, pp. 2224–2228.
[70] X. Z. Gao and S. J. Ovaska, "Motor fault detection using Elman neural network with genetic algorithm-aided training," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2000, vol. 4, pp. 2386–2392.
[71] G. V. Puskorius and L. A. Feldkamp, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 279–297, Mar. 1994.
[72] S. Wu and T. W. S. Chao, "Induction machine fault detection using SOM-based RBF neural networks," *IEEE Trans. Ind. Electron.*, vol. 51, no. 1, pp. 183–194, Feb. 2004.

Bimal K. Bose (S'59–M'60–SM'78–F'89–LF'96) received the B.E. degree from Bengal Engineering College (currently Bengal Engineering and Science University), Calcutta, India, in 1956, the M.S. degree from the University of Wisconsin, Madison, in 1960, and Ph.D. degree from Calcutta University, Calcutta, in 1966.

He held the Condra Chair of Excellence (Endowed Chair) in Power Electronics at the University of Tennessee, Knoxville, since 1987, where he was responsible for teaching and the research program in power electronics and motor drives. Concurrently, he served as Distinguished Scientist (1989–2000) and Chief Scientist (1987–1989) of EPRI-Power Electronics Applications Center, Knoxville. Prior to this, he was a Research Engineer in the General Electric Corporate Research and Development Center (now GE Global Research Center), Schenectady, NY, for 11 years (1976–1987), an Associate Professor of Electrical Engineering, Rensselaer Polytechnic Institute, Troy, NY, for five years (1971–1976), and a faculty member at Bengal Engineering and Science University for 11 years (1960–1971). He is specialized in power electronics and motor drives, specially including power converters, PWM techniques, microcomputer/DSP control, electric/hybrid vehicle drives, renewable energy systems, and artificial intelligence (expert system, fuzzy logic, and neural network) applications in power electronic systems. He has been power electronics consultant in a large number of industries. He holds Honorary Professorship with Shanghai University (1991), China University of Mining and Technology (1995), Xi'an Mining University (1998), Huazhong University of Science and Technology (2002), and Honorary Adviser of Beijing Power Electronics R&D Center (1990). He has authored/edited seven books in power electronics: *Power Electronics and Motor Drives – Advances and Trends* (New York: Academic Press, 2006), *Modern Power Electronics and AC Drives* (Englewood Cliffs, NJ: Prentice-Hall, 2002), *Power Electronics and AC Drives* (Englewood Cliffs, NJ: Prentice-Hall, 1986), *Power Electronics and Variable Frequency Drives* (New York: Wiley, 1997), *Modern Power Electronics* (Piscataway, NJ: IEEE Press, 1992), *Microcomputer Control of Power Electronics and Drives* (Piscataway, NJ: IEEE Press, 1987), and *Adjustable Speed AC Drive Systems* (Piscataway, NJ: IEEE Press, 1981). He has given tutorials, keynote addresses, and invited seminars extensively throughout the world, particularly in IEEE sponsored programs and conferences. He has authored more than 190 papers and holds 21 U.S. patents.

Dr. Bose is a recipient of a number of awards, including the IEEE Power Electronics Society Newell Award (2005), IEEE Meritorious Achievement Award in Continuing Education (1997), IEEE Lamme Gold Medal (1996), IEEE-IES Eugene Mittelmann Award (for lifetime achievement) (1994), IEEE Region 3 Outstanding Engineer Award (1994), IEEE-IAS Outstanding Achievement Award (1993), Calcutta University Mouat Gold Medal (1970), GE Silver Patent Medal (1986), GE Publication Award (1985), and a number of IEEE prize paper awards. He also received the Distinguished Alumnus Award (2006) from Bengal Engineering and Science University. He has served the IEEE in various capacities, including Chairman of the IEEE Industrial Electronics Society (IES) Power Electronics Council, Associate Editor of the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, IEEE-IECON Power Electronics Chairman, Chairman of the IEEE Industry Applications Society (IAS) Industrial Power Converter Committee, IAS member of the Neural Network Council, Vice-Chair of IEEE Medals Council, Member of IEEE Energy Policy Committee, Member of the IEEE Fellow Committee, Member of Lamme Medals Committee, etc. He has been a Member of the Editorial Board of the PROCEEDINGS OF THE IEEE since 1995. He was the Guest Editor of the PROCEEDINGS OF THE IEEE (Special Issue on Power Electronics and Motion Control), August 1994. He has served as a Distinguished Lecturer of both the IAS and IES, and is now Vice-Chairman of the IAS Distinguished Lecturer Program.