A Stator-Flux-Oriented Vector-Controlled Induction Motor Drive With Space-Vector PWM and Flux-Vector Synthesis by Neural Networks

João O. P. Pinto, *Student Member, IEEE*, Bimal K. Bose, *Life Fellow, IEEE*, and Luiz Eduardo Borges da Silva, *Member, IEEE*

***Abstract*—A stator-flux-oriented vector-controlled induction motor drive is described where the space-vector pulsewidth modulation (SVM) and stator-flux-vector estimation are implemented by artificial neural networks (ANNs). ANNs, when implemented by dedicated hardware application-specific integrated circuit chips, provide extreme simplification and fast execution for control and feedback signal processing functions in high-performance ac drives. In the proposed project, a feedforward ANN-based SVM, operating at 20 kHz sampling frequency, generates symmetrical pulsewidth modulation (PWM) pulses in both undermodulation and overmodulation regions covering the range from dc (zero frequency) up to square-wave mode at 60 Hz. In addition, a programmable cascaded low-pass filter (PCLPF), that permits dc offset-free stator-flux-vector synthesis at very low frequency using the voltage model, has been implemented by a hybrid neural network which consists of a recurrent neural network (RNN) and a feedforward neural network (FFANN). The RNN-FFANN-based flux estimation is simple, permits faster implementation, and gives superior transient performance when compared with a standard digital-signal-processor-based PCLPF. A 5-hp open-loop volts/Hz-controlled drive incorporating the proposed ANN-based SVM and RNN-FFANN-based flux estimator was initially evaluated in the frequency range of 1.0–58 Hz to validate the performance of SVM and the flux estimator. Next, the complete 5-hp drive with stator-flux-oriented vector control was evaluated extensively using the PWM modulator and flux estimator. The drive performance in both volts/Hz control and vector control were found to be excellent.**

***Index Terms*—Induction motor, neural network, space-vector pulsewidth modulation, vector control.**

Paper IPCSD 01–035, presented at the 2000 Industry Applications Society Annual Meeting, Rome, Italy, October 8–12, and approved for publication in the IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS by the Industrial Drives Committee of the IEEE Industry Applications Society. Manuscript submitted for review May 1, 2000 and released for publication June 28, 2001. This work was supported in part by General Motors Advanced Technology Vehicles (GMATV) and CAPES of Brazil.

J. O. P. Pinto is with the Department of Electrical Engineering, University of Tennessee, Knoxville, TN 37996-2100 USA, on leave from the Universidade Federal do Mato Grosso do Sul, Campo Grande, MS, Brazil (e-mail: jpinto@utk.edu).

B. K. Bose is with the Department of Electrical Engineering, University of Tennessee, Knoxville, TN 37996-2100 USA (e-mail: bbose@utk.edu).

L. E. B. da Silva is with the Department of Electronics, Federal School of Engineering of Itajuba, Itajuba, MG 37500-903 Brazil (e-mail: leborges@irr.efei.br).

Publisher Item Identifier S 0093-9994(01)08327-X.

I. INTRODUCTION

SENSORLESS vector control and neural-network-based intelligent control are recently receiving wide attention in the literature. Sensorless control eliminates speed, flux, and torque sensors and replaces them by digital-signal-processor (DSP)-based estimation using the machine terminal voltages and currents. Thus, the cost of the drive is reduced and reliability is enhanced. However, the estimation algorithm tends to be complex particularly at very low frequency because of integration offset and parameter variation problems. The stator-flux-oriented direct vector control [1] is attractive from the parameter variation point of view because the stator resistance is the only parameter under consideration, which can be compensated somewhat easily [2]. However, integration of low-voltage low-frequency signals for flux estimation becomes difficult because a single-stage integrator with large time constant tends to build up dc offset at the integrator output. Among several methods proposed in the literature, the programmable cascaded low-pass filter (PCLPF) technique [3] looks very promising for solving this problem.

Space-vector pulsewidth modulation (SVM) has recently become a very popular pulsewidth modulation (PWM) method for voltage-fed converter ac machine drives because of its superior harmonic quality and extended linear range of operation. However, one difficulty of SVM is that it requires complex online computation that usually limits its operation up to several kilohertz of switching frequency. Of course, switching frequency can be extended by using a high-speed DSP and simplified algorithms including lookup tables. Lookup tables, unless very large, tend to reduce pulsewidth resolution. Power semiconductor switching speed, particularly for an insulated gate bipolar transistor (IGBT), is improving dramatically in recent years. DSP-based SVM is especially suitable for drives where the switching frequency is usually restricted below 10 kHz.

Both the PCLPF and SVM have been implemented by neural networks [4], [5]. Neural networks, in general, are showing very high promise for simplification of control and feedback signal processing in high-performance sensorless vector-controlled drives. The additional advantages of neural network are fast parallel computation, learning capability, and fault tolerance, which are not possible by DSP-based computation. Neural networks have already been demonstrated

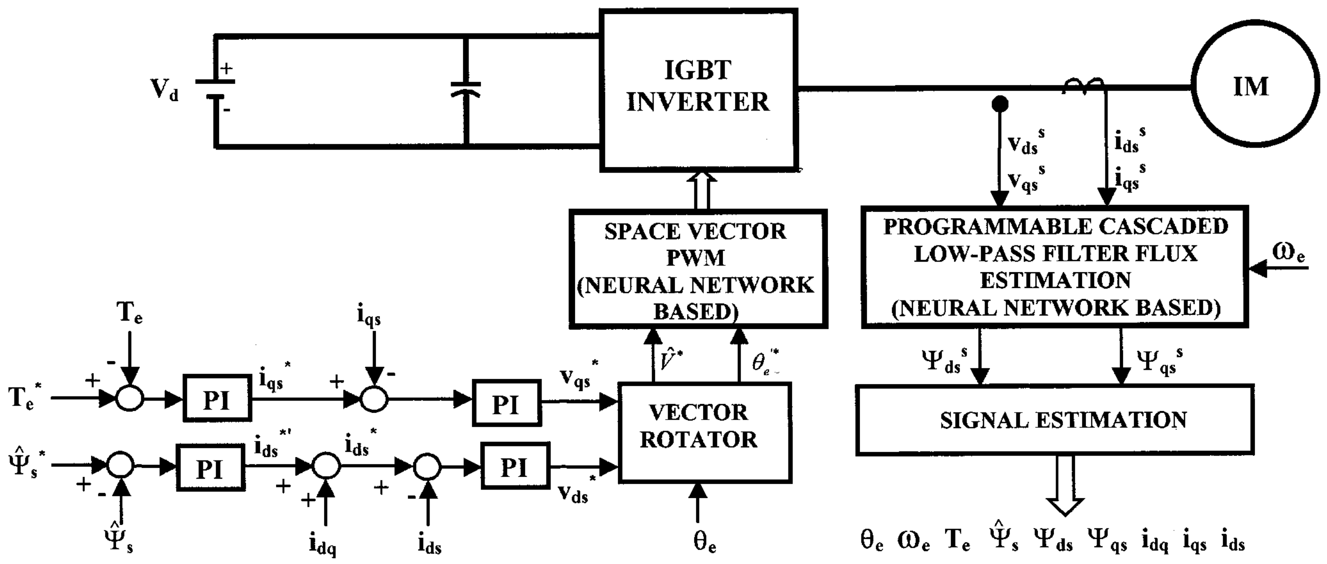


Fig. 1. Stator-flux-oriented vector control with neural-network-based SVM and flux estimation.

for individual functions, such as vector transformation, PWM, flux- and unit-vector estimation, speed and torque estimation, intelligent control, fault diagnosis and fault tolerant control, etc. It appears that, in the future, more and more functions will be integrated in a single neural network and, ultimately, one or two neural application-specific integrated circuits (ASIC) chips will be able to cover all the control and feedback signal processing functions of a high-performance drive. This will provide extreme cost advantage and reliability improvement of drive systems. Currently, most of the neural networks applied in power electronic systems have feedforward multilayer-perceptron-type geometry. They are easier to train but can only perform a static input–output nonlinear mapping function. The emulation of a dynamical system with temporal behavior is often needed in many control and feedback estimation functions of a high-performance drive. The capability of a feedforward network to handle a dynamical system by incorporating time delays in feedback and feedforward elements is limited. On the other hand, recurrent neural networks (RNNs) or feedback neural networks are very efficient to solve such problems, but their training is difficult.

This paper describes a stator-flux-oriented vector-controlled drive incorporating the neural-network-based SVM and PCLPF. Neural network implementation of the SVM and PCLPF were described before [5], [4]. However, PCLPF implementation has been simplified by using a perceptron-type ANN instead of a polynomial-type ANN (PNN). Both the ANN-based SVM and PCLPF were incorporated in the drive and their performance was validated by open-loop volts/Hz drive control. Then, the performance of the complete vector-controlled drive was evaluated using the above networks.

II. System Description

Fig. 1 shows the simplified block diagram of the stator-flux-oriented vector-controlled drive system under consideration. An electric-vehicle-type drive system has been considered. However, the control is perfectly general and is valid for any industrial drive system. All the symbols are standard in the figure. The power circuit consists of a battery, filter capacitor, and three-phase two-level IGBT PWM inverter that generates variable-voltage variable-frequency power for the machine. The drive has torque and stator-flux control in the outer loops. In the present system, the flux command $\hat{\psi}_s^*$ is constant, and the drive operates in the constant-torque region only. However, operation can be easily extended to the field-weakening constant power region by programming the flux as a function of speed. The torque control loop has an internal i_{qs} current control loop. The speed control loop can be easily added over the torque loop, if desired. The flux loop output is added with the decoupling compensation current i_{dq} to establish the i_{ds}^* current command. The synchronous current control loops then generate the v_{qs}^* and v_{ds}^* signals as shown in the figure. These signals are then vector rotated with the help of unit vector angle θ_e to generate the voltage vector magnitude $\hat{V}^*$ and orientation angle $\theta_e'^*$ as follows:

$$\hat{V}^* = \sqrt{(v_{qs}^*)^2 + (v_{ds}^*)^2} \tag{1}$$

$$\theta_e'^* = \theta_e + \tan^{-1}\left(\frac{v_{qs}^*}{v_{ds}^*}\right) \tag{2}$$

where θ_e is the orientation angle between the d^s and d^e axes. The system incorporates two neural-network-based functions as indicated. These are space-vector PWM controller and PCLPF-based flux estimator. The PWM controller receives the V^* and θ_e' signals at the input and translates to gate drive signals for the IGBT inverter.

The machine terminal voltages and currents are sensed, filtered by hardware low-pass filters and then converted (not shown) into stationary frame $d^s - q^s$ signals before A/D conversion. These signals are then converted to stationary frame

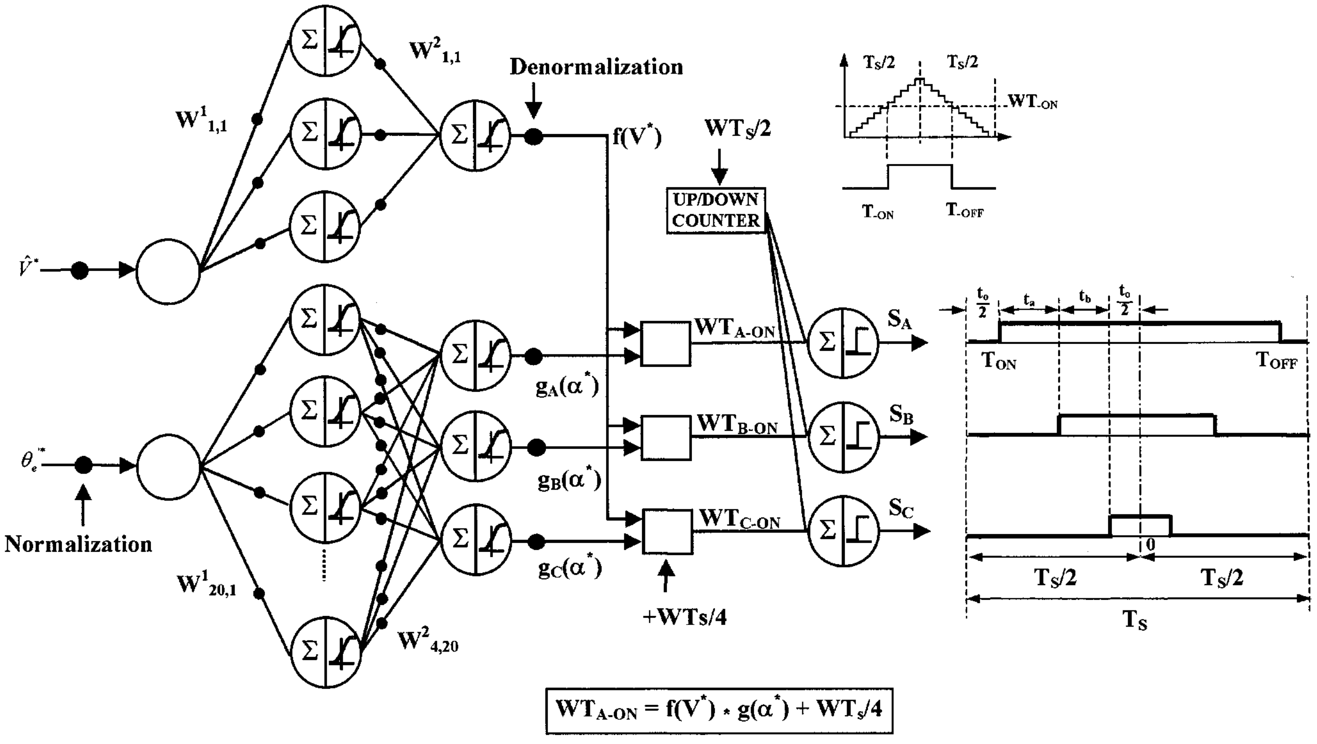


Fig. 2. Topology of neural-network-based SVM modulator.

flux-vector signals by neural-network-based PCLPF using the following:

$$\psi^s_{ds} = \int (v^s_{ds} - i^s_{ds}R_s)\,dt \tag{3}$$

$$\psi^s_{qs} = \int (v^s_{qs} - i^s_{qs}R_s)\,dt. \tag{4}$$

The signal estimation block does the following computations [3]:

$$\hat{\psi}_s = \sqrt{(\psi^s_{ds})^2 + (\psi^s_{qs})^2} \tag{5}$$

$$\theta_e = \sin^{-1}\left(\frac{\psi^s_{qs}}{\hat{\psi}_s}\right) \tag{6}$$

$$i_{ds} = i^s_{qs}\cos\theta_e - i^s_{ds}\sin\theta_e \tag{7}$$

$$i_{qs} = i^s_{qs}\sin\theta_e + i^s_{ds}\cos\theta_e \tag{8}$$

$$i_{dq} = \frac{\sigma L_s i^2_{qs}}{\psi_{ds} - \sigma L_s i_{ds}} \tag{9}$$

$$\psi_{ds} = \psi^s_{qs}\cos\theta_e - \psi^s_{ds}\sin\theta_e \tag{10}$$

$$\psi_{qs} = \psi^s_{qs}\sin\theta_e + \psi^s_{ds}\cos\theta_e \tag{11}$$

$$T_e = \frac{3P}{4}[\psi_{ds}i_{qs} - \psi_{qs}i_{ds}] \tag{12}$$

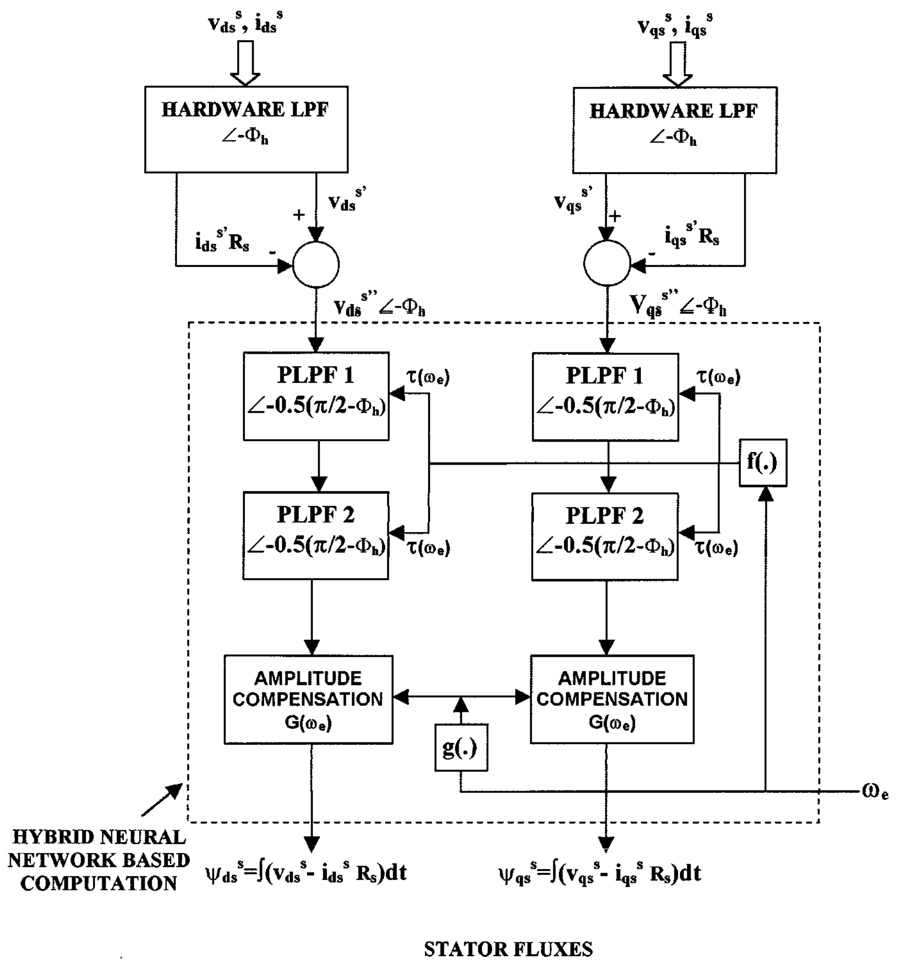
$$\omega_e = \frac{[(v^s_{qs} - i^s_{qs}R_s)\psi^s_{ds} - (v^s_{ds} - i^s_{ds}R_s)\psi^s_{qs}]}{\hat{\psi}^2_s} \tag{13}$$

where ω_e is the frequency (rad/s), i_{ds}, i_{qs} are the rotating frame currents, ψ_{ds}, ψ_{qs} are the rotating frame stator fluxes, and all other signals are in standard notation. Note that the stator flux $\hat{\psi}_s$ is oriented to the d^e axis in stator-flux-oriented vector control.

III. Neural-Network-Based SVM

Neural-network-based SVM by an indirect method (using a single timer) has been described in [5]. However, it will be briefly reviewed here for completeness of the paper. The SVM algorithms were initially developed for undermodulation and overmodulation regions and the corresponding expressions for turn-on and turn-off times were derived. However, for neural network implementation, these algorithms were unified with simplification and expressed in graphical form where the turn-on time (T_{ON}) of an IGBT could be plotted as a function of angle θ_e for the six $\pi/3$ angle sectors. This modification indicated that the voltage magnitude and the angle can be handled in parallel and then combined at the end in order to generate digital words which can be converted to pulsewidths through a timer. Fig. 2 shows the topology of the neural-network-based SVM. It receives the $\hat{V}^*$ and $\theta_e'^*$ signals at the input and generates symmetrical pulses for three phases at the output, as shown in the figure. Basically, the network consists of two subnets: the magnitude subnet (shown in the upper part) which implements the function $f(\hat{V}^*)$ which is linear in the undermodulation region but nonlinear in the overmodulation region; and the angle subnet implements the pulsewidth function $g(\alpha^*)$ for the three phases at phase shift angles of $2\pi/3$. Note that the sigmoidal activation functions of the angle subnet generate only unipolar outputs, and these are converted to bipolar outputs by adding a fixed bias in the denormalization process. The digital words corresponding to turn-on time (T_{ON}) of the three phases are given as

$$W_{T_{A\text{-ON}}} = f(V^*) * g_A(\alpha^*) + \frac{WT_s}{4} \tag{14}$$



v_{ds}^s, i_{ds}^s
v_{qs}^s, i_{qs}^s
HARDWARE LPF
$\angle$-Φh
$v_{ds}^{s'}$
$i_{ds}^{s'} R_s$
$v_{qs}^{s'}$
$i_{qs}^{s'} R_s$
$v_{ds}^{s''} \angle -\Phi_h$
$V_{qs}^{s''} \angle -\Phi_h$
PLPF 1
PLPF 2
$\angle -0.5(\pi/2-\Phi_h)$
$\tau(\omega_e)$
f(.)
g(.)
AMPLITUDE COMPENSATION G(ωe)
ωe
HYBRID NEURAL NETWORK BASED COMPUTATION
$\psi_{ds}^s = \int(v_{ds}^s - i_{ds}^s R_s)dt$
$\psi_{qs}^s = \int(v_{qs}^s - i_{qs}^s R_s)dt$
STATOR FLUXES

Fig. 3. PCLPF for stator-flux vector estimation.

$$W_{T_B\text{-ON}} = f(V^*) * g_B(\alpha^*) + \frac{WT_s}{4} \tag{15}$$

$$W_{T_C\text{-ON}} = f(V^*) * g_C(\alpha^*) + \frac{WT_s}{4} \tag{16}$$

where T_s is the sampling time ($T_s = 1/f_{sw}$, f_{sw} is the switching frequency), $g(\alpha^*)$ is the pulsewidth function of a phase at unit voltage amplitude, and $f(\hat{V}^*)$ is the voltage magnitude function. For the symmetrical pulsewidth of each phase, the turn-off time can be given in the form

$$T_{\text{OFF}} = T_s - T_{\text{ON}}. \tag{17}$$

The pulsewidth signals for the three phases were generated by using a single UP/DOWN counter, as shown in the figure. Both the subnets were trained and tested extensively for accurate matching with the training data generated by simulation. The backpropagation algorithm of the MATLAB Neural Network Tool Box was used for offline training of the network. The magnitude subnet was trained with data at a 1.0 V increment of $\hat{V}^*$, and takes 1750 epochs (with error below 1%). The angle subnet was trained with angle increment of 2.16° in the interval of 0°–360°. The training takes 4800 epochs (with error below 0.5%). Once the network was trained and tested thoroughly, it was operated at 20-kHz sampling frequency ($T_s = 50\ \mu$s) in the present drive system.

IV. Hybrid RNN-Based Flux-Vector Synthesis

A. Cascaded Low-Pass Filter Flux-Vector Estimation

The stator flux vectors ψ_{ds}^s and ψ_{qs}^s were estimated by integrating the corresponding stator voltages behind the stator resistance drops, i.e., by solving (3) and (4). Two identical stages of PCLPFs, as shown in Fig. 3, were used for the estimation of ψ_{ds}^s and ψ_{qs}^s where the portion implemented by the hybrid neural network is indicated by the dotted enclosure. The PCLPF essentially permits ideal integration of voltage from extremely low frequency to high-frequency field-weakening range without building dc offset at the output. Basically, it uses a series of n (the figure shows $n = 2$) first-order programmable (or adaptive) low-pass filters in order to obtain a total phase shift of 90° and an appropriate gain in order to have ideal integration at any frequency. The time constant (τ) of the component filters and amplitude compensation gain (G) of the PCLPF are a nonlinear function of frequency. The programmable τ keeps the phase shift and gain of each filter stage identical at a predetermined value at any frequency, and the amplitude compensation gain G adjusts the total gain of the

PCLPF in order to achieve ideal integration of stator voltage behind the stator resistance drop. The hardware low-pass filter in the front end introduces lagging phase-shift angle φ_h (function of frequency) which is compensated in each stage of the filter, as indicated in the figure. If, for example, $\phi_h = 0$, the component filter phase shift angle remains constant at 45° at any frequency. This compensation feature permits generous filtering of signals by hardware filter without introducing error in the integration.

B. Hybrid Neural Network Implementation

Although a large number of PCLPF stages (n) are desirable for ideal integration, in this section we will consider only two stages $(n = 2)$ for neural network implementation because of computational complexity. Each low-pass filter stage in Fig. 3 can be discretized and expressed by transfer function with Z transform in the form

$$\frac{Y(z)}{U(z)} = \left(\frac{K}{z-a}\right). \tag{18}$$

The two filter stages in cascade can, therefore, be expressed in the form

$$\frac{Y_1(z)}{U(z)} = \left(\frac{Kz^{-1}}{1-az^{-1}}\right) \tag{19}$$

$$\frac{Y_2(z)}{Y_1(z)} = \left(\frac{Kz^{-1}}{1-az^{-1}}\right) \tag{20}$$

where the gain G is merged equally as $\sqrt{G}$ in each filter stage. The corresponding discrete-time equations are

$$y_1(k) = ay_1(k-1) + Ku(k-1) \tag{21}$$

$$y_2(k) = ay_2(k-1) + Ky_1(k-1) \tag{22}$$

where $y_1(k)$ is the output of the first stage, $y_2(k)$ is the output of the second stage, $u(k)$ is the filter input, $a = (1 - (T_s/\tau))$ (T_s is the sampling time), and $K = (T_s\sqrt{G})/\tau$. Equations (21) and (22) can be expressed in matrix form as

$$\begin{bmatrix} y_1(k+1) \\ y_2(k+1) \end{bmatrix} = \begin{bmatrix} a & 0 \\ K & a \end{bmatrix} \begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} + \begin{bmatrix} K \\ 0 \end{bmatrix} u(k). \tag{23}$$

The equivalent RNN will be represented by the following equation:

$$\begin{bmatrix} y_1(k+1) \\ y_2(k+1) \end{bmatrix} = \begin{bmatrix} W_{11} & 0 \\ W_{21} & W_{22} \end{bmatrix} \begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} + \begin{bmatrix} W_{13} \\ 0 \end{bmatrix} u(k). \tag{24}$$

The upper part of Fig. 4 shows the topology of the RNN where W_{11}, W_{21}, W_{22}, and W_{13} are the network weights. Basically, it consists of two identical second-order RNNs or feedback networks for synthesis of each of ψ_{ds}^s and ψ_{qs}^s flux components. This type of RNN was first proposed by Williams and Zipser in 1989 [6] and is often known as a real-time recurrent network. The network has linear activation functions at the output. Although $W_{11} = W_{22} = a$ and $W_{21} = W_{13} = K$ in (24), all the weights have been considered as independent variables and a function of frequency. The programmable weights (as a function of frequency) are generated by a separate feedforward multilayer-perceptron- or backpropagation-type network shown in the lower part of Fig. 4. The RNN was trained offline by an extended-Kalman-filter (EKF)-based algorithm [7] which was developed in the laboratory in the MATLAB environment. The example data for training was generated by simulation. Although the fundamental frequency range in the present drive system is narrow (0–60 Hz), it was decided to train the hybrid network in the range of 0.01–200 Hz with the step size of 1.0 Hz. At each frequency, 201 input–output data pairs were used for training. Once the network is trained, it can easily interpolate at any fraction of a hertz. At every frequency, the four weights W_{11}, W_{21}, W_{22}, and W_{13}, shown in the figure, are tuned so that the input voltage wave and the corresponding flux output wave match precisely with an error less than 10^{-5}. The training time depends on the frequency and the sampling time, and it is typically 1 h for an input frequency with a Pentium-II (200 MHz)-based PC and sampling time of 125 μs. The table of weights (in the form of curves) as a function of frequency is shown at the right of Fig. 4. For example, the trained weights at 32 Hz (200.96 rad/s) are: $W_{11} = 0.9764$, $W_{22} = 0.9739$, $W_{21} = 0.0254$, and $W_{13} = 0.0002$. These weights, as a function of frequency (basically, a lookup table), were generated by the feedforward network shown in the lower part of the figure. The bias network is not shown for simplicity. The three hidden-layer neurons use sigmoidal-type nonlinear activation function whereas the four output-layer neurons use linear transfer function. The network was trained with 1.0-Hz step size to generate the vector of four weights needed for the RNN at the same frequency. Only 130 epochs were needed to train this part by backpropagation algorithm, and 30 trials were made to reach to the smallest size of the network. All the weights in the matrix form are given as follows:

nput layer

$$W^1 = \begin{bmatrix} W_{1.1}^1 \\ W_{2.1}^1 \\ W_{3.1}^1 \end{bmatrix} = \begin{bmatrix} -1.5972 \\ -3.3231 \\ 15.7331 \end{bmatrix} \tag{25}$$

hidden layer

$$W^2 = \begin{bmatrix} W_{1.1}^2 & W_{1.2}^2 & W_{1.3}^2 \\ W_{2.1}^2 & W_{2.2}^2 & W_{2.3}^2 \\ W_{3.1}^2 & W_{3.2}^2 & W_{3.3}^2 \\ W_{4.1}^2 & W_{4.2}^2 & W_{4.3}^2 \end{bmatrix} = \begin{bmatrix} 1.1890 & 0.0009 & 0.0003 \\ -0.0003 & 0.0000 & 0.0000 \\ 1.0798 & -0.1113 & 0.0256 \\ -1.1591 & 0.0276 & -0.0064 \end{bmatrix}. \tag{26}$$

The input signals of both the networks were normalized, and signals at the output of the RNN were denormalized.

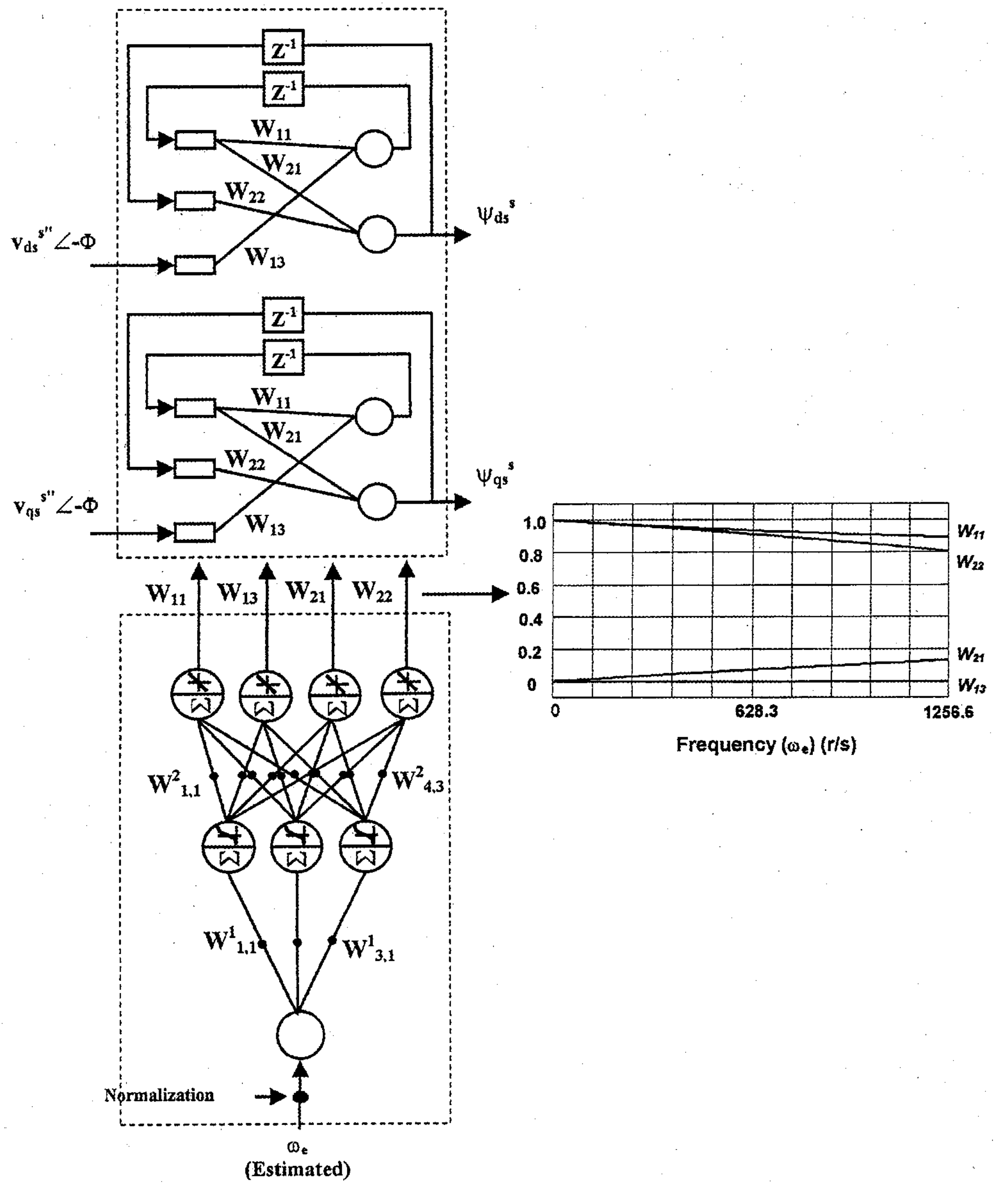


Fig. 4. Hybrid RNN for stator flux estimation.

C. RNN Training Algorithm by EKF

The training of the feedforward neural network by the gradient-descent backpropagation algorithm is somewhat familiar to the readers. However, the RNN training, particularly by EKF algorithm, is generally not a familiar technique. Therefore, in this section, we will briefly review this powerful algorithm.

Recently, the EKF algorithm has shown significant merits for training both feedforward and RNNs [7]. It has been shown that the training data and total training time can be substantially lessened by this algorithm. Computationally, this algorithm is simpler than gradient-descent algorithms although it requires the same derivative information. The EKF algorithm does not require batch processing of data and, therefore, it is very suitable for online training.

The EKF algorithm is well known for parameter and state estimation problems in the presence of noise. The RNN training to determine the unknown weights can also be viewed as a parameter estimation problem. Here, the problem involves computation of derivatives of the network outputs with respect to the trainable weights which are to be tuned. The training algorithm is formulated as a weighted least-square minimization problem, where the error vector is the difference between the functions of the network's output nodes and the desirable values of these functions. The desirable or target vector at time k is given by

$$d(k) = [d_1(k) \cdots d_N(k)]^T \tag{27}$$

where the vector is of length N. Let the output $y(k)$ of a network be represented by the vector $h(k)$ of the same length so that the error vector is

$$\xi(k) = d(k) - h(k). \tag{28}$$

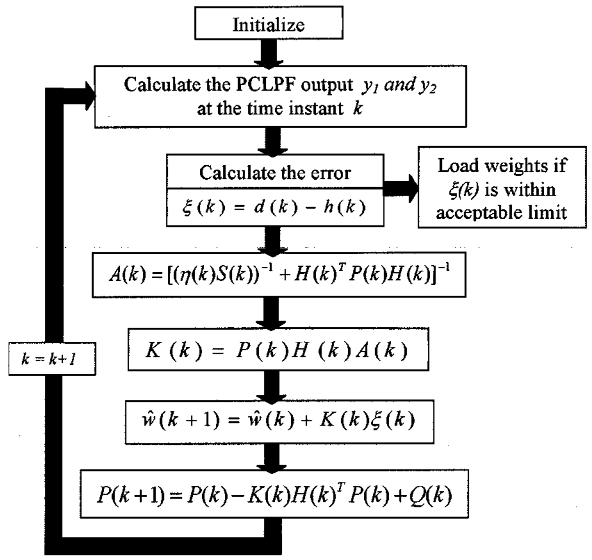

Initialize
Calculate the PCLPF output y_1 and y_2 at the time instant k
Calculate the error
$\xi(k) = d(k) - h(k)$
Load weights if $\xi(k)$ is within acceptable limit
$A(k) = [(\eta(k)S(k))^{-1} + H(k)^T P(k)H(k)]^{-1}$
$K(k) = P(k)H(k)A(k)$
$\hat{w}(k+1) = \hat{w}(k) + K(k)\xi(k)$
$P(k+1) = P(k) - K(k)H(k)^T P(k) + Q(k)$
$k = k+1$

Fig. 5. Computation flowchart for EKF training algorithm of RNN.

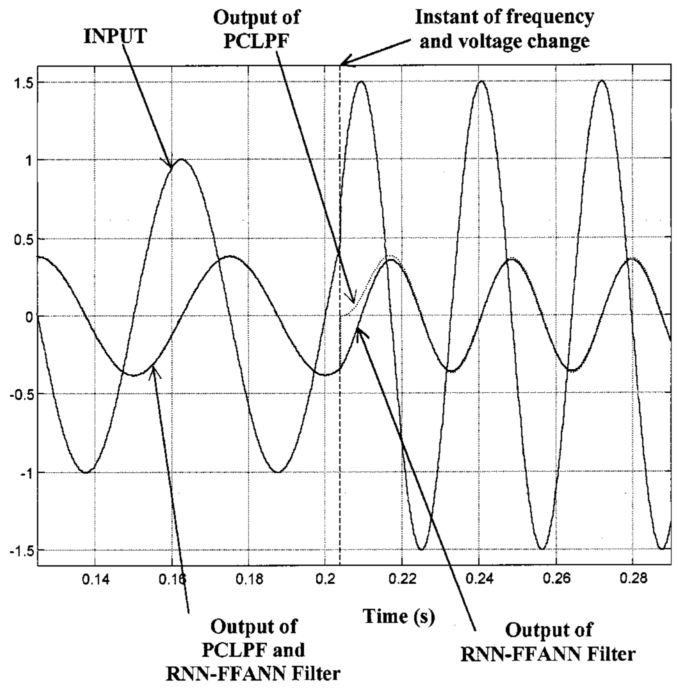


Fig. 6. Comparison of response of RNN-MLPNN filter and PCLPF.

The neural network training cost function $E(k)$ is then given by

$$E(k) = \frac{1}{2}\xi(k)^T S(k)\xi(k) \tag{29}$$

where $S(k)$ a user-specified nonnegative definite weighting matrix. The RNN trainable weights can be arranged into an M-dimensional vector $W(k)$. The EKF algorithm updates the network weights and the approximate error covariance matrix $P(k)$ which models the correlation between each pair of weights in the network.

TABLE I
DRIVE SYSTEM PARAMETERS

DC supply voltage (V_d) :	300 V
Induction motor:	5 hp, 230 V, 4-pole, Reliance Electric NEMA-Class B
	Frequency range: 0 – 60 Hz
	Power factor (full load): 0.85
	Efficiency (full load): 86%
	Stator resistance (R_s): 0.5814 Ω
	Rotor resistance (R_r) : 0.4165 Ω
	Stator leak. inductance (L_{ls}): 3.479 mH
	Rotor leak. inductance (L_{lr}): 4.15 mH
	Magnetizing inductance (L_m): 78.25 mH
	Rotor inertia (J) : 0.1 Kg.m^2
	Fan load [$T_L = K\,\omega_r^2$] : K = 8.25 X10^{-5}
	Base frequency (ω_{base}): 377 rad/s
	Rated torque (T_r): 19.8 Nm

The algorithm works in such a way that at discrete time k, the input signals and recurrent nodes outputs are propagated through the network, and the function $h(k)$ is calculated. The error vector $\xi(k)$ is computed and evaluated for the current estimates of weights W. The dynamic derivatives of each component of $h(k)$ are formed with respect to the weights of the network. These derivatives are arranged in the form of an $M \times N$ matrix $H(k)$ as follows:

$$H(k) = \begin{bmatrix} \frac{\partial(net_1)}{\partial W_1} & \cdots & \frac{\partial(net_N)}{\partial W_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial(net_1)}{\partial W_m} & \cdots & \frac{\partial(net_N)}{\partial W_m} \end{bmatrix} \tag{30}$$

where $net_1 \cdots net_N$ = respective neuron output. Then, $W(k)$ and $P(k)$ are updated using the following EKF recursion equations:

$$A(k) = [(\eta(k)S(k))^{-1} + H(k)^T P(k)H(k)]^{-1} \tag{31}$$
$$K(k) = P(k)H(k)A(k) \tag{32}$$
$$\hat{w}(k+1) = \hat{w}(k) + K(k)\xi(k) \tag{33}$$
$$P(k+1) = P(k) - K(k)H(k)^T P(k) + Q(k) \tag{34}$$

where $\eta(k)$ is the scalar learning parameter and $Q(k)$ a diagonal covariance matrix that provides a mechanism to attenuate the noise affecting the signals involved in the training process. The presence of the $Q(k)$ matrix helps in avoiding numerical divergence of the algorithm and also eliminates stopping at local minima.

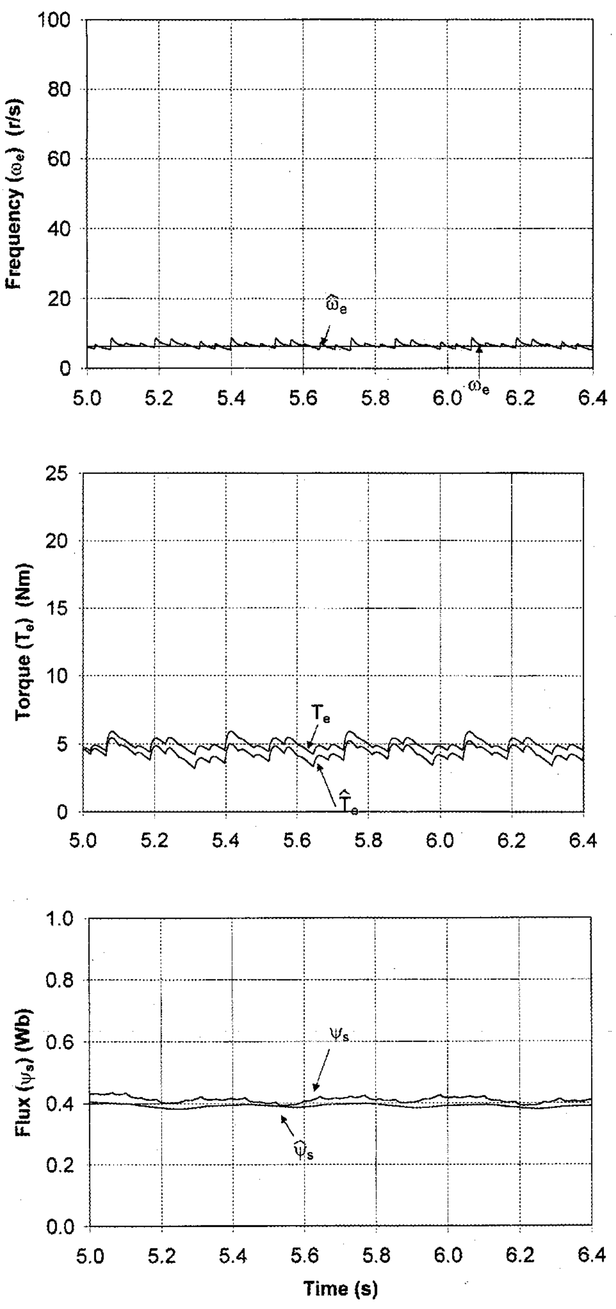


Fig. 7. Open-loop volts/Hz control performance and feedback signal estimation (1.0 Hz).

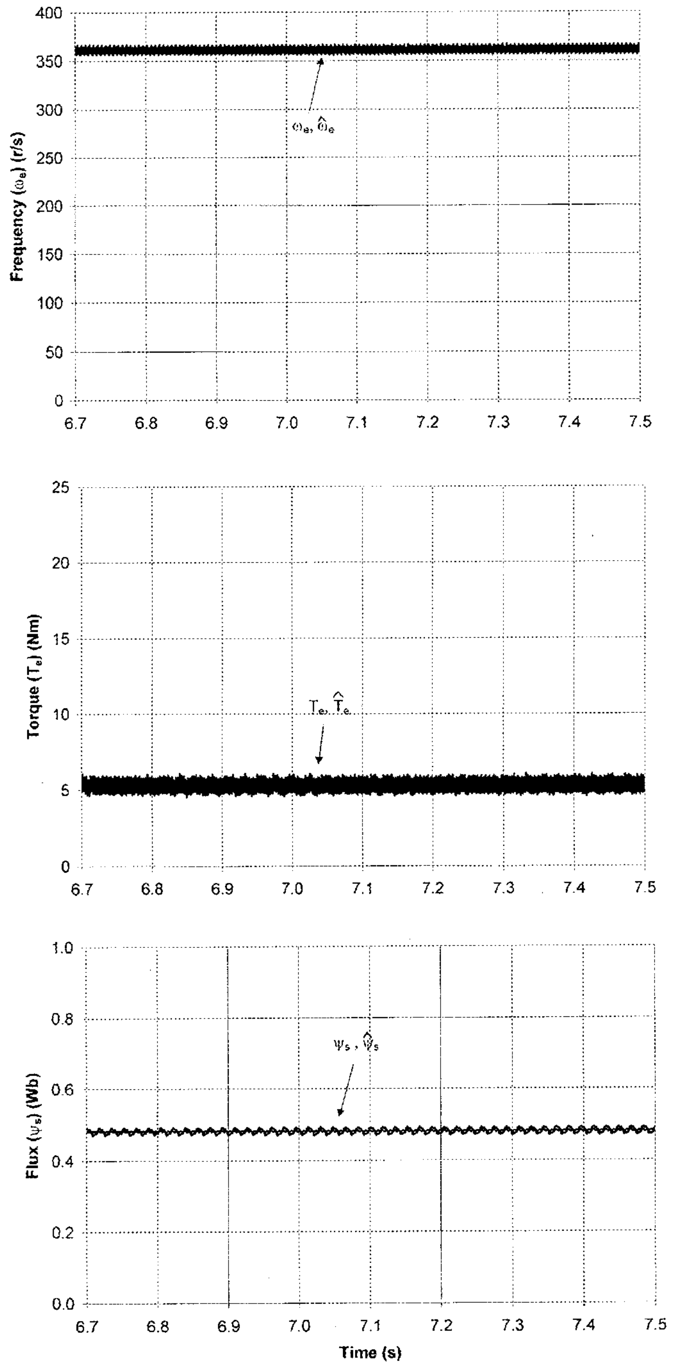


Fig. 8. Open-loop volts/Hz control performance and feedback signal estimation (58 Hz).

D. RNN Training Procedure of PCLPF

In the previous section, the general EKF algorithm was presented for training of a RNN. We will now discuss the application of the algorithm for the two-stage PCLPF described previosly.

Defining $net_1 = y_1$ and $net_2 = y_2$ in Fig. 4, we can write (24) in the form

$$net_1 = W_{11}y_1(k-1) + W_{13}u(k-1) \tag{35}$$

$$net_2 = W_{21}y_1(k-1) + W_{22}y_2(k-1). \tag{36}$$

The partial derivatives of net_1 and net_2 with respect to the weights are given as

$$\begin{aligned} \frac{\partial(net_1)}{\partial W_{11}} &= y_1(k-1) & \frac{\partial(net_1)}{\partial W_{21}} &= 0 \\ \frac{\partial(net_1)}{\partial W_{12}} &= 0 & \frac{\partial(net_1)}{\partial W_{13}} &= u(k-1) \\ \frac{\partial(net_2)}{\partial W_{11}} &= 0 & \frac{\partial(net_2)}{\partial W_{21}} &= y_1(k-1) \\ \frac{\partial(net_2)}{\partial W_{22}} &= y_2(k-1) & \frac{\partial(net_2)}{\partial W_{21}} &= 0. \end{aligned} \tag{37}$$

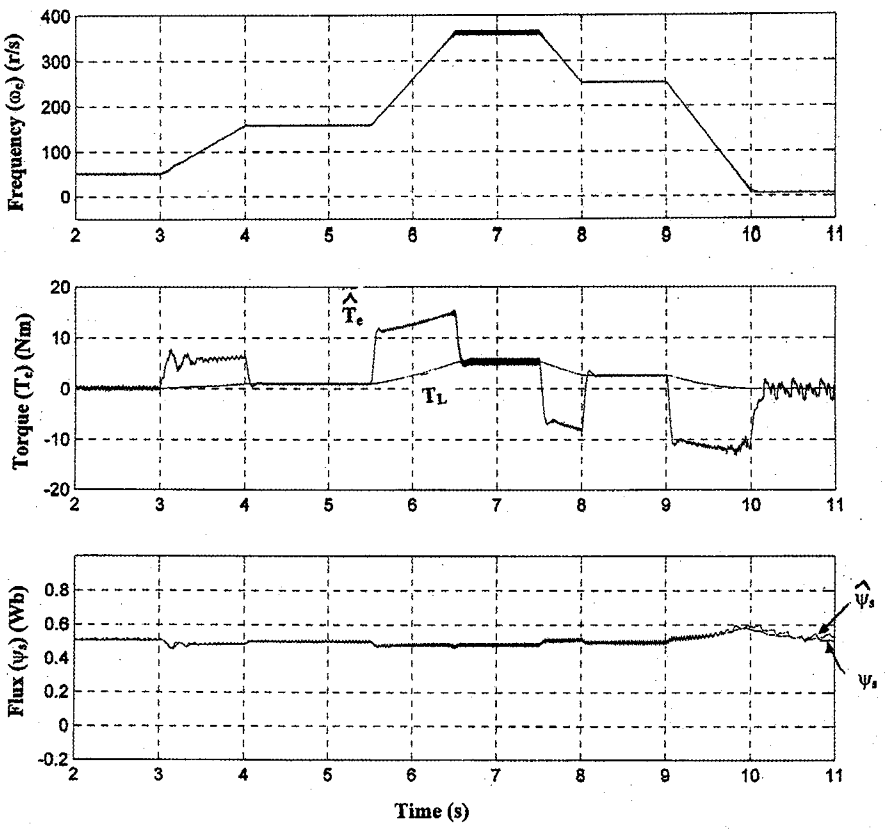


Fig. 9. Dynamic performance with volts/Hz control.

Therefore, the $H(k)$ matrix in (30) can be expressed as

$$H(k) = \begin{bmatrix} y_1(k-1) & 0 \\ 0 & y_1(k-1) \\ 0 & y_2(k-1) \\ u(k-1) & 0 \end{bmatrix}. \tag{38}$$

At this point, the only missing information needed to run the EKF algorithm is the error vector which can be calculated from the desired filter output and actual RNN output, i.e., $\xi(k) = d(k) - h(k)$ where

$$d(k) = \begin{bmatrix} y_{1d}(k) \\ y_{2d}(k) \end{bmatrix} \quad \text{and} \quad h(k) = \begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix}. \tag{39}$$

All the information needed to run the recursive set of equations (31)–(34) related to the EKF algorithm are now available. Fig. 5 shows the sequence of computations in the EKF algorithm for training the PCLPF.

Fig. 6 shows the response of the hybrid ANN and compares it with the DSP-based estimation of the PCLPF. The responses are identical at steady state but show superiority in the transient response. In the practical case, the jump is reasonably small.

V. System Performance Evaluation

Once the neural networks for SVM and flux estimation were trained and tested extensively offline, these were integrated in an open-loop volts/Hz-controlled 5-hp drive and performance was evaluated in the frequency range of 1.0–58 Hz by simulation. Table I shows the parameters of the drive system. One of the objectives for open-loop volts/Hz control is to validate all the feedback signal estimation before closing the loops in the vector-controlled drive. The ANN-based SVM was operated at 20 kHz ($Ts = 50\ \mu$s), as mentioned before, and the remainder of the system had a sampling time of 125 μs. Fig. 7 shows the steady-state performance with the command frequency of 1.0 Hz (6.28 rad/s). The estimated frequency $(\hat{\omega}_e)$, torque $(\hat{T}_e)$, and stator flux $(\hat{\psi}_s)$ are reasonably ripple free, and the accuracy was typically within 7%, 6%, and 2.5%, respectively. The accuracy and ripple were excellent at the higher frequency linear modulation range. Fig. 8 shows the corresponding performance at a frequency of 58 Hz (overmodulation region) where the accuracy of the estimated signals was much superior. Fig. 9 shows the drive performance in dynamic condition with fan load (see the parameters in Table I) where the frequency is ramped up and down in several steps within the frequency ranges of 8 Hz (50.24 rad/s)–58 Hz (364.24 rad/s) and 58–1.0 Hz (6.28 rad/s), respectively. The estimated torque and stator flux show some amount of underdamping at low frequency, but it disappears at higher frequency. The open-loop flux with constant volts/Hz control shows change at higher current and low frequency due to stator resistance drop, as expected. Once the performance was satisfactory in open-loop volts/Hz control with full validation of feedback signal estimation, the complete drive system with stator-flux-oriented vector control (Fig. 1) was evaluated thoroughly by simulation for both steady-state and

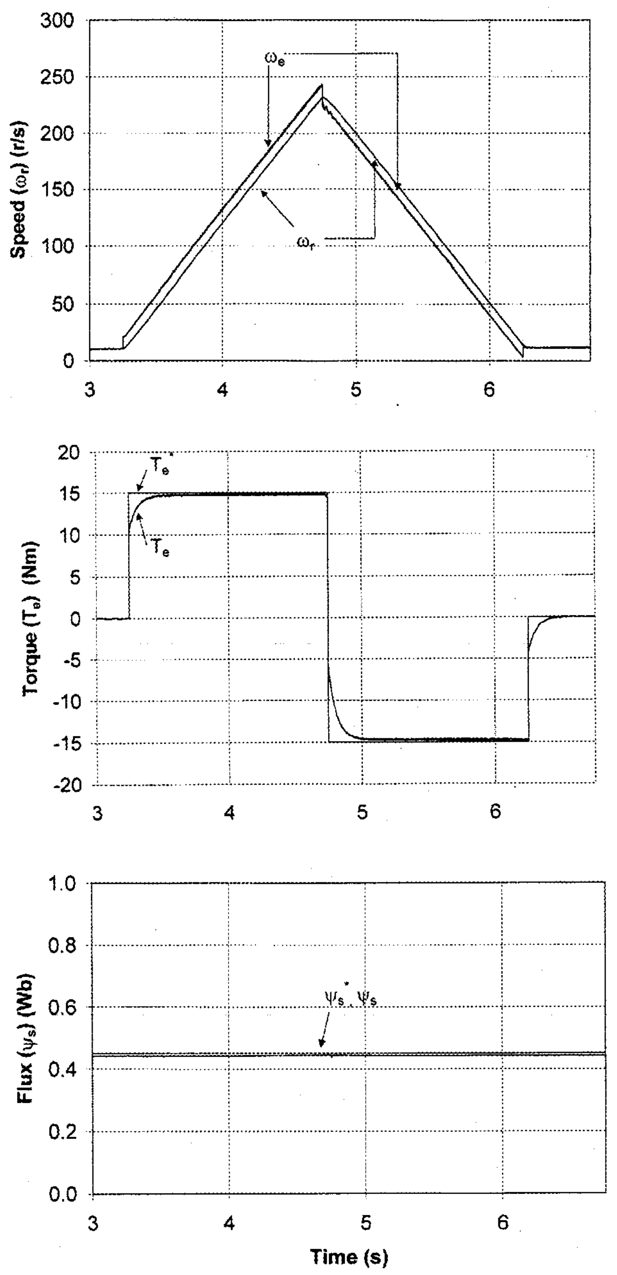


Fig. 10. Stator-flux-oriented vector control performance at acceleration and deceleration.

dynamic conditions. Fig. 10 shows the drive performance with pure inertia load in the full cycle of acceleration and deceleration at rated flux in the constant torque region. With the constant torque command, the electrical speed (ω_r) was increased linearly from 6.28 rad/s (60 r/min) to 230 rad/s (2200 r/min) and then brought down linearly to the initial speed by constant torque regeneration. The performance in the whole region was found to be excellent.

VI. Conclusion

This paper has described a stator-flux-oriented vector-controlled induction motor drive that incorporates a high-frequency feedforward-neural-network-based SVM modulator and a hybrid-neural-network-based stator flux estimator. The hybrid ANN consists of an RNN where the weights are updated by a feedforward network. The recurrent ANN has been trained by the EKF algorithm whereas the feedforward ANNs are trained by the backpropagation algorithm. Since the EKF training algorithm is not generally familiar to readers, a brief description has been included. The drive with outer loop torque control is suitable for a propulsion-type drive, but it can be easily extended to an industrial drive with speed control loop. The control does not consider zero-speed startup condition. Except for the proportional–integral (P–I) control loops in Fig. 1, the whole drive control system can be implemented by three ANN chips, i.e., PWM modulator, flux estimator, and signal estimator described in [8]. Currently, implementation of P–I control loops by RNN is in progress. It appears that full control and feedback estimation of a high-performance drive system are possible by a few neural network chips in the near future and, ultimately, one chip will perform all the functions.

References

[1] X. Xu, R. N. De Donker, and D. W. Novotny, “A stator flux oriented induction motor drive,” in *Proc. IEEE PESC’88*, 1988, pp. 870–876.
[2] B. K. Bose and N. R. Patel, “Quasifuzzy estimation of stator resistance of induction motor,” *IEEE Trans Power Electron.*, vol. 13, pp. 401–409, May 1998.
[3] ——, “A sensorless stator flux oriented vector controlled induction motor drive with neuro-fuzzy based performance enhancement,” in *Conf. Rec. IEEE-IAS Annu. Meeting*, 1997, pp. 393–400.
[4] L. E. Borges, 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, pp. 662–665, June 1999.
[5] J. O. P. Pinto, B. K. Bose, L. E. Borges, and M. P. Kazmierkowski, “A neural network based space vector PWM controller for voltage-fed inverter induction motor drive,” *IEEE Trans. Ind. Applicat.*, vol. 36, pp. 1628–1636, Nov./Dec. 2000.
[6] S. Haykin, *Neural Networks*. New York: Macmillan, 1994.
[7] G. V. Puskorius and L. A. Feldkamp, “Neurocontrol of nonlinear dynamic systems with Kalman filter trained recurrent networks,” *IEEE Trans. Neural Networks*, vol. 5, pp. 279–297, Mar. 1994.
[8] M. G. Simoes and B. K. Bose, “Neural network based estimation of feedback signals for a vector controlled induction motor drive,” *IEEE Trans. Ind. Applicat.*, vol. 31, pp. 620–629, May/June 1995.
[9] B. K. Bose, Ed., *Power Electronics and Variable Frequency Drives*. New York: IEEE Press, 1997.
[10] B. K. Bose, *Modern Power Electronics and AC Drives*. Upper Saddle River, NJ: Prentice-Hall, 2001.

João O. P. Pinto (S’97) received the B.S. degree from the Universidade Estadual Paulista, Ilha Solteira, Brazil, and the M.S. degree from the Universidade Federal de Uberlândia, Uberlândia, Brazil, in 1990 and 1993, respectively. He is currently working toward the Ph.D. degree at the University of Tennessee, Knoxville.

He has been an Assistant Professor at the Universidade Federal do Mato Grosso do Sul, Campo Grande, Brazil, since 1994. His research interests include neural networks, fuzzy logic, genetic algorithms, and wavelet applications to power electronics, PWM techniques, drives, and electric machines control.

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

He currently holds the Condra Chair of Excellence in Power Electronics at the University of Tennessee, Knoxville, where he has been active in organizing the power electronics program for the last 14 years. He was the Distinguished Scientist and Chief Scientist of the EPRI-Power Electronics Applications Center, Knoxville. Early in his career, he served as a faculty member at Calcutta University for 11 years. In 1971, he joined Rensselaer Polytechnic Institute, Troy, NY, as an Associate Professor of Electrical Engineering. In 1976, he joined General Electric Corporate Research and Development, Schenectady, NY, as a Research Engineer and served there for 11 years. He has been a Consultant to more than ten industries. His research interests spread over the whole spectrum of power electronics, and specifically include power converters, ac drives, microcomputer control, EV drives, and artificial intelligence techniques in power electronics and drives. He has authored more than 150 published papers and is the holder of 21 U.S. patents. He is the author/editor of six books: *Modern Power Electronics and AC Drives* (Upper Saddle River, NJ: Prentice-Hall, 2001), *Power Electronics and AC Drives* (Englewood Cliffs, NJ: Prentice-Hall, 1986), *Adjustable Speed AC Drive Systems* (New York: IEEE Press, 1981), *Microcomputer Control of Power Electronics and Drives* (New York: IEEE Press, 1987), *Modern Power Electronics* (New York: IEEE Press, 1992), and *Power Electronics and Variable Frequency Drives* (New York: IEEE Press, 1997). He has served in numerous national and international conferences and professional organizations. He has given keynote addresses, seminars, and tutorials all over the world.

Dr. Bose has served the IEEE in various capacities that include Chairman of the Power Electronics Council of the IEEE Industrial Electronics Society (IES), Associate Editor of IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, IEEE-IECON Power Electronics Chairman, Chairman of the Industrial Power Converter Committee of the IEEE Industry Applications Society (IAS), IAS member in Neural Network Council, and various other professional committees. He has been a member of the Editorial Board of the PROCEEDINGS OF THE IEEE since 1995. He has served as a Distinguished Lecturer of the IAS and IES. He was the Guest Editor of the PROCEEDINGS OF THE IEEE Special Issue on Power Electronics and Motion Control (August 1994). He has received a GE Publication Award, Silver Patent Medal, and several IEEE Prize Paper Awards. He received the IAS Outstanding Achievement Award "for outstanding contributions in the application of electricity to industry" (1993), IES Eugene Mittelmann Award "in recognition of outstanding contributions to research and development in the field of power electronics and a lifetime achievement in the area of motor drive" (1994), IEEE Region 3 Outstanding Engineer Award "for outstanding achievements in power electronics and drives technology" (1994), IEEE Lamme Gold Medal "for contributions in power electronics and drives (1996), IEEE Continuing Education Award "for exemplary and sustained contributions to continuing education (1997), and IEEE Millennium Medal (2000). For his research contributions, he was awarded the Premchand Roychand Scholarship and Mouat Gold Medal by Calcutta University in 1968 and 1970, respectively.

Luiz Eduardo Borges da Silva (S'84–M'89) was born in Passa Quatro, Brazil. He received the B.S. degree in electrical engineering and the M.S. degree from the Escola Federal de Engenharia de Itajubá, Itajubá, Brazil, and the Ph.D. degree in power electronics from the Ecole Polytechnique de Montreal, Montreal, QC, Canada, in 1977, 1982, and 1988, respectively.

From 1988 to 1992, he was the Head of the Electronic Department, Escola Federal de Engenharia de Itajubá. During 1998, he held a post-doctoral position in the Department of Electrical Engineering, University of Tennessee, Knoxville. He currently teaches power electronics and digital control systems at the Escola Federal de Engenharia de Itajubá, where he is also with the Power Electronics Group. His main research interests are applications of artificial intelligence in power electronics and signal processing.