Direct Self Control of Induction Motor Based on Neural Network

K. L. Shi, T. F. Chan, *Member IEEE*, Y. K. Wong, *Senior Member, IEEE*, and S. L. Ho

***Abstract*—This paper presents an artificial-neural-network-based direct-self-control (ANN–DSC) scheme for an inverter-fed three-phase induction motor. In order to cope with the complex calculations required in direct self control (DSC), the proposed artificial-neural-network (ANN) system employs the individual training strategy with fixed-weight and supervised models. A computer simulation program is developed using Matlab/Simulink together with the Neural Network Toolbox. The simulated results obtained demonstrate the feasibility of ANN–DSC. Compared with the classical digital-signal-processor-based DSC, the proposed ANN-based scheme incurs much shorter execution times and, hence, the errors caused by control time delays are minimized.**

***Index Terms*—Direct self control, induction motor drive, Matlab/Simulink, neural networks.**

I. Introduction

THE neural network is well known for its learning ability and approximation to any arbitrary continuous function [1]. Recently, it has been proposed in the literature that neural networks can be applied to parameter identification and state estimation of induction motor control systems [2]. However, artificial-neural-network (ANN) vector control of induction motors is seldom reported. One of the reasons is the complexity of the controller.

Direct self control (DSC) [4], [5] is a dynamic, recurrent, and nonlinear signal-processing method that theoretically can give an inverter-fed three-phase induction motor an excellent performance [3]–[5]. Since complicated calculations are involved, it is difficult to implement DSC using common integrated-circuit (IC) hardware. In the classical DSC system, the control algorithms are realized by serial computation on a digital-signal-processor (DSP) board. As a predictive control scheme, however, DSC has a steady-state control error caused by the time delays required for the lengthy computations, the actual values depending largely on the control algorithm and hardware performance [6]. As a result, an upper limit has to be imposed on the switching frequency of the inverter-fed motor drive. Present-day power electronic devices, on the other hand, are capable of very fast switching operations. In order to fully exploit the advantages of these devices, the control time delays need to be minimized. The neural network, with its simple architecture and inherent parallel computation capability, offers a promising alternative to realization of a high-performance DSC drive. In the future, the neural-network controller may be implemented by application-specific integrated-circuit (ASIC) chips. The electrically trainable analog neural network (ETANN) chip (Intel 80 170NX) had a performance adequate for drive applications, taking only 3 μs to process through each layer [7]. This chip may serve as a benchmark for comparison purpose as modern neural devices are likely to have comparable or higher processing speeds.

This paper presents an ANN algorithm that can be used in place of the DSP serial calculations in a DSC system. Following a brief introduction of neural networks, the paper gives a detailed explanation of their use in the realization of the computational modules of the control process. Simulation results will also be presented to demonstrate the feasibility of the ANN-based DSC.

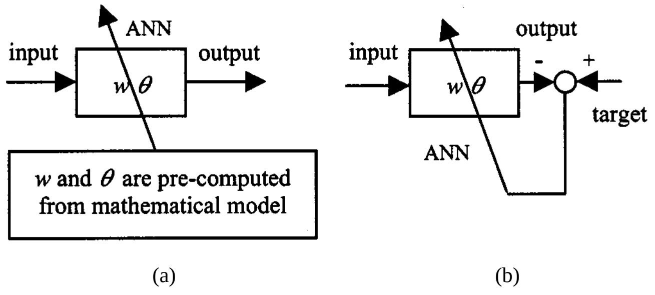


Fig. 1. Neural networks using different training algorithms. (a) Fixed-weight network. (b) Supervised network.

II. Neural Networks

In general, a neural model is mathematically represented by a basis function (net function) and an activation function (neuron function). The selection of these functions often depends on the applications of the neural network. In other words, application-driven neural models are only loosely tied to the biological realities. Linear basis function $u(w_i, x)$ is a hyperplane-type

Paper IPCSD 01–034, 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 July 9, 2001. This work was supported by The Hong Kong Polytechnic University under Research Grant V157.

K. L. Shi was with the Department of Electrical Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong, China and was also with Honeywell Engineering and Systems, Honeywell Aerospace Canada, Mississauga, ON, L5L 3S6 Canada. He is now with the Electrical and Computer Engineering Department, Ryerson Polytechnic University, Toronto , ON M5B 2K3 Canada.(e-mail: shikeli@sympatico.ca).

T. F. Chan, Y. K. Wong, and S. L. Ho are with the Department of Electrical Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: eetfchan@polyu.edu.hk; eeykwong@polyu.edu.hk; eeslho@polyu.edu.hk).

Publisher Item Identifier S 0093-9994(01)08313-X.

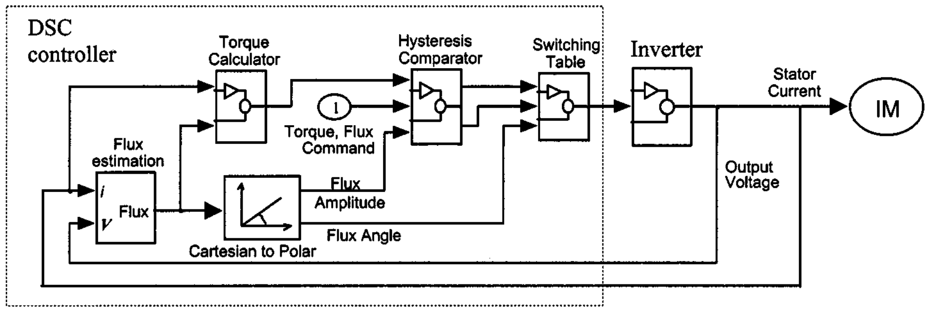


Fig. 2. Construction of the DSC system in Matlab/Simulink window.

function, where w_i stands for the weight matrix, x for the input vector, and θ_i for the bias or threshold. Mathematically,

$$u_i(w_i, x) = \sum_{j=1}^{n} w_{ij}x_j + \theta_i \tag{1}$$

where j is the dimension of input.

The net value as expressed by the basis function $u_i(w_i, x)$ will be immediately transformed by an activation function of the neuron. Thus,

$$y_i = f(u_i) \tag{2}$$

where y_i is the net output and $f(.)$ is the activation function.

The memory of a neural network lies in weights and biases. The neural networks can be classified, in terms of how the weights and biases are obtained, into three categories [8]. They are: 1) fixed-weight; 2) unsupervised; and 3) supervised networks. In this paper, the fixed-weight networks and supervised networks are used. The constructions of the two networks are shown in Fig. 1. The training data consist of pairs of input and target produced by the DSC mathematical model.

The characteristic of the fixed-weight network is that the weights and biases are precomputed and pre-stored from training data. The fixed-weight network, which is also called the "direct design method," can be used to implement an exact mathematical model. In same cases, its implementation is easier than the supervised network.

In the supervised network, the weights and biases are adaptively trained by a learning mechanism, which has been the mainstream of neural model development. The back-propagation learning rule [8] is used to design the supervised networks for the DSC, details of which are presented below.

It is convenient to regard the threshold θ as an extra weight, i.e., $\theta = w_{n+1}$ [8]. The net value given by (1) can be rewritten as

$$u_i(w_i, x) = \sum_{j=1}^{n+1} w_{ij}z_j = W_iZ \tag{3}$$

where $W_i = [w_{i1}\ w_{i2}\ \cdots\ w_{in}\ \theta_i]$ and $Z = [x_1\ x_2\ \cdots\ x_n\ 1]^T$.

The sum squared error E (cost function) for the set of M patterns of input is given by [2]

$$E = \frac{1}{2}\sum_{m=1}^{M} E_m = \tfrac{1}{2}\sum_{m=1}^{M}\sum_{i=1}^{I}(t_i^m - y_i^m) \tag{4}$$

where

E_m	squared error of the output layer;
I	dimension of the output vector;
y^m	actual output vector;
t^m	corresponding desired output vector.

The weights are changed to reduce the cost function E to a minimum value by the gradient descent method.

The best initial weights and biases for back-propagation networks are created at random utilizing the minimum and maximum values of each input. The jth weight-update equation of the ith neuron is given as

$$w_{ij}(t+1) = w_{ij}(t) + \eta\left(\frac{\partial E_m}{\partial w_{ij}(t)}\right) \tag{5}$$

where

η	learning rate;
$w_{ij}(t+1)$	new weight;
$w_{ij}(t)$	old weight.

The training strategies may be divided as mutual (whole) and individual (local) [8]. In mutual training, the training of all the weights is influenced by all the input/output values. In individual training, the training of an individual subnet will not be influenced by the inputs and outputs of other subnets. Pure mutual training is almost impossible for DSC due to three reasons. Firstly, the direct self controller is a dynamic (there are integrators), recurrent (there are hysteresis comparators), and nonlinear system. Secondly, the eight input variables (V_a, V_b, V_c, i_a, i_b, i_c, T^*, $|\lambda^*|$) constitute a huge training set. Thirdly, it may take substantially more iterations to reach a mutual agreement between all the nodes. For simpler and faster design, the individual training strategy is adopted in this paper.

III. Neural-Network-Based DSC

A DSC scheme consists typically of 3/2 transformations of current and voltage, flux estimation, torque calculation, flux angle encoder, flux magnitude computation, hysteresis comparator, and optimum switching table. Fig. 2 shows a DSC system in the Matlab/Simulink window, which consists of a DSC controller, an inverter, and an induction motor [9].

Based on DSC principle, the neural-network controller is divided into the following five sub-nets, which are individually trained:

1) flux estimation sub-net (supervised) with dynamic neurons;

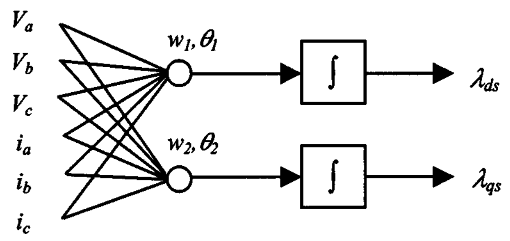


Fig. 3. Dynamic linear net for flux estimation.

2) torque calculation sub-net (fixed weight) with square neurons;
3) flux angle encoder and magnitude calculation sub-net (supervised and fixed weight) with logsig neurons and tansig neurons;
4) hysteresis comparator sub-net (fixed weight) with recurrent neurons;
5) optimum switching table sub-net (supervised) with hard limit neurons.

A. Flux Estimation Sub-Net

The flux estimation may be expressed as [5], [9]

$$\dot{\lambda}_{ds} = (V_a - R_s i_a) - \frac{1}{2}(V_b - R_s i_b) - \frac{1}{2}(V_c - R_s i_c) \quad (6)$$

$$\dot{\lambda}_{qs} = \frac{\sqrt{3}}{2}(V_b - R_s i_b) - \frac{\sqrt{3}}{2}(V_c - R_s i_c) \quad (7)$$

where V_a, V_b, V_c are phase voltages and i_a, i_b, i_c are phase currents, which are obtained from the voltage and current sensors.

Neuron models can be divided into two basic types: namely, static and dynamic. A dynamic neuron is one whose output is described by a differential equation [10]. Hence, the flux estimation sub-net should be constructed by the two dynamic neurons that consist of linear neurons and integrators. A supervised method, *viz.* the back-propagation learning rule, is used to train the linear neurons until they can approximate (6) and (7).

Fig. 3 shows the flux estimation network. Using a random generator function of Matlab, ten random inputs of the vector $[V_a, V_b, V_c, i_a, i_b, i_c]$ are produced. Using these random inputs, the target outputs can be obtained from (6) and (7). Since the network is linear, convergence can be obtained in relatively few training epochs. For the induction motor being studied, the weights and biases have been obtained, as shown in the equations at the bottom of this page.

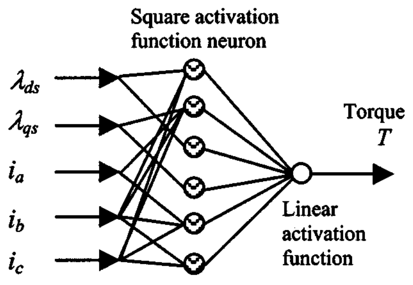


Fig. 4. Neural network for torque calculation.

B. Torque Calculation Sub-Net

The torque equation for a DSC system is given by

$$T = \frac{P}{2}\frac{2}{3}\left(\lambda_{ds} i_{qs} - \lambda_{qs} i_{ds}\right) \quad (8)$$

where P is the number of motor poles.

Since there are four inputs, i.e., λ_{ds}, λ_{qs}, i_{ds}, and i_{qs}, the data of all training patterns will be huge if high precision is required. To avoid the training difficulties, the fixed-weight method is adopted. Equation (8) may be rewritten as a sum of square functions

$$T = \frac{P}{6}\left[(\lambda_{ds} + i_{qs})^2 - (\lambda_{qs} + i_{ds})^2 - \lambda_{ds}^2 + \lambda_{qs}^2 + i_{ds}^2 - i_{qs}^2\right] \quad (9)$$

where

$$\begin{bmatrix} i_{ds} \\ i_{qs} \end{bmatrix} = \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}.$$

A two-layer fixed-weight neural network is used to implement (9) directly, as shown in Fig. 4. The first layer is a square activation function with the weight and bias w_1 and θ_1, while the second layer is a linear active function with the weight and bias w_2 and θ_2 as follows:

$$w_1 = \begin{bmatrix} 1 & 0 & 0 & \sqrt{3}/2 & -\sqrt{3}/2 \\ 0 & 1 & 1 & -1/2 & -1/2 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1/2 & -1/2 \\ 0 & 0 & 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix}$$

$$w = \begin{bmatrix} 0.5725 & -0.2910 & 0.7790 & -0.3740 & -0.7496 & -0.0172 \\ -0.3899 & 1.3073 & -0.2469 & 0.4546 & -0.7029 & 0.4575 \end{bmatrix}$$

$$\theta = \begin{bmatrix} -0.2029 \\ -0.2836 \end{bmatrix}$$

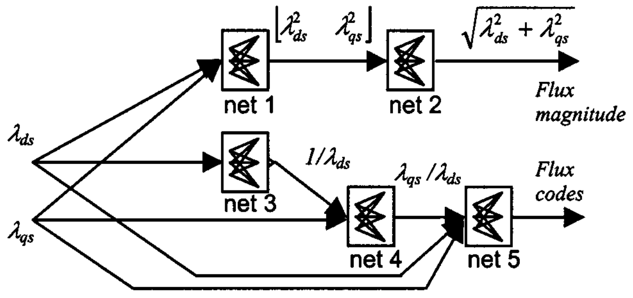

$[\lambda_{ds}^2 \;\; \lambda_{qs}^2]$
$\sqrt{\lambda_{ds}^2+\lambda_{qs}^2}$
net 1
net 2
Flux magnitude
λ_{ds}
$1/\lambda_{ds}$
$\lambda_{qs}/\lambda_{ds}$
Flux codes
net 3
λ_{qs}
net 4
net 5

Fig. 5. Individual training scheme for flux angle and magnitude calculations in Matlab window.

$$\theta_1 = \begin{bmatrix} 0\\0\\0\\0\\0\\0 \end{bmatrix}$$

$$w_2 = [1 \quad -1 \quad -1 \quad 1 \quad 1 \quad -1]$$
$$\theta_2 = [0].$$

C. Flux Angle Encoder and Flux Magnitude Calculation Sub-Net

The flux angle α and flux magnitude f can be calculated from the flux space vectors λ_{ds} and λ_{qs}. The flux angle is then encoded as $B_1B_2B_3$ as follows:

$$f = \sqrt{\lambda_{ds}^2 + \lambda_{qs}^2} \tag{10}$$

$$\alpha = \tan^{-1}\left(\lambda_{ds}/\lambda_{qs}\right) \tag{11}$$

$$B_1B_2B_3 = \text{encoder}\,(\alpha). \tag{12}$$

In order to obtain more accurate results and to simplify the design, (10)–(12) are rewritten as

$$v = \left[\lambda_{ds}^2 \quad \lambda_{qs}^2\right] \tag{13}$$

$$f = \sqrt{v \times \begin{bmatrix}1\\1\end{bmatrix}} \tag{14}$$

$$\xi = 1/\lambda_{qs} \tag{15}$$

$$\zeta = \lambda_{ds} \times \xi = \left[(\lambda_{ds}+\xi)^2 - \lambda_{ds}^2 - \xi^2\right]\Big/2 \tag{16}$$

$$B_1B_2B_3 = \text{encoder}\,(\zeta). \tag{17}$$

The network of flux angle encoder and flux magnitude calculation consists of five nets, as shown in Fig. 5, i.e., net 1 with two square neurons implements (13), net 2 with four tansig neurons implements (14), net 3 with four logsig neurons, net 4 with three square neurons, and net 5 with ten hard limit neurons implement (15)–(17), respectively. Net 1, net 4, and net 5 are designed using the fixed-weight method. Net 2 and net 3 are designed using the supervised method. The output layers of net 1–net 4 will be merged with the input layers of their next sub-nets according to the rule shown in Fig. 14. Hence, weights and biases of the output layers of these nets have not been listed hereinafter.

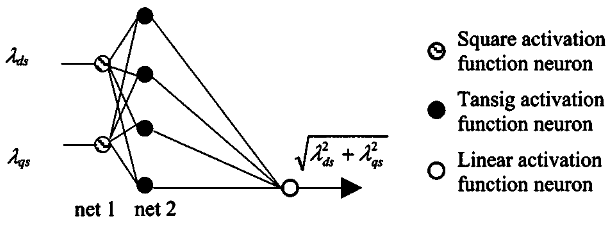

λ_{ds}
λ_{qs}
$\sqrt{\lambda_{ds}^2+\lambda_{qs}^2}$
net 1
net 2
Square activation function neuron
Tansig activation function neuron
Linear activation function neuron

Fig. 6. Implementation of the flux magnitude calculation.

Design of net 1 is similar to that used for the torque calculation sub-net described in Section III-B. Weight w and bias θ of net 1 with two square neurons can be directly designed as follows according to (13):

$$w(\text{net}\,1) = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \quad \theta(\text{net}\,1) = \begin{bmatrix}0\\0\end{bmatrix}.$$

The back-propagation learning rule is used to train net 2. Five-hundred input/output pairs for training net 2 are produced by the square root function. After 50 000 training epochs, the sum-squared error E is less than 0.01. Weight w and bias θ of net 2 with tansig neurons is as follows:

$$w(\text{net}\,2) = \begin{bmatrix}-0.1781 & -0.1781\\2.5008 & 2.5008\\-2.9325 & -2.9325\\0.5233 & 0.5233\end{bmatrix} \quad \theta(\text{net}\,2) = \begin{bmatrix}1.5851\\6.0920\\-6.2367\\0.3963\end{bmatrix}.$$

Fig. 6 shows the implementation of the flux magnitude calculation using net 1 and net 2.

The back-propagation learning rule is used to train net 3 until they can approximate the reciprocal function. One-thousand input/output pairs are produced by the reciprocal function to train net 3. After 100 000 training epochs, the sum-squared error E is less than 0.02

$$w(\text{net}\,3) = \begin{bmatrix}-31.6560\\2.8829\\14.8182\\47.8563\end{bmatrix} \quad \theta(\text{net}\,3) = \begin{bmatrix}-0.7218\\-0.0022\\0.0207\\0.7360\end{bmatrix}.$$

With three square neurons, net 4 implements (16) using the same technique described in Section III-B as follows:

$$w(\text{net}\,4) = \begin{bmatrix}1/2 & 1/2\\-1/2 & 0\\0 & -1/2\end{bmatrix} \quad \theta(\text{net}\,4) = \begin{bmatrix}0\\0\\0\end{bmatrix}.$$

Net 5 implements the flux angle encoding directly from $\lambda_{ds}/\lambda_{qs}$ (output of net 4), λ_{ds}, and λ_{qs}. To improve the algorithm, the trigonometric function computations of flux angle, which are necessary in previous DSC schemes, are replaced by logic operations. With reference to Fig. 7, the flux angle

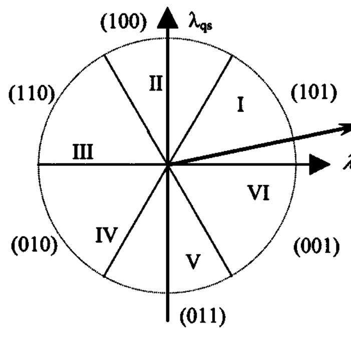


Fig. 7. Space flux encode ($B_1\, B_2\, B_3$).

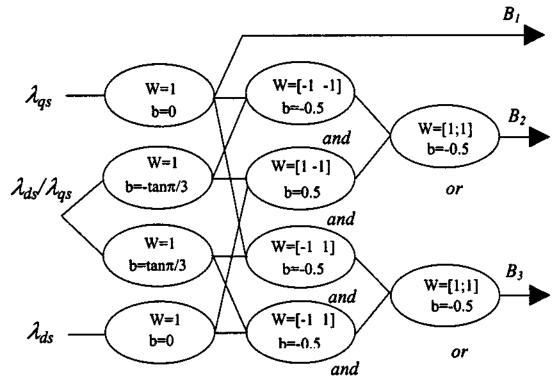


Fig. 8. Flux angle encoder (net 5) with hard limit neurons.

code ($B_1\, B_2\, B_3$) can be directly derived from the following equations:

$$B_1 = \begin{cases} 1, & \lambda_{qs} \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

$$B_2 = \begin{cases} 1, & (\lambda_{qs}/\lambda_{ds} \geq -\tan\pi/3 \text{ and } \lambda_{ds} < 0) \\ & \text{or } (\lambda_{qs}/\lambda_{ds} < -\tan\pi/3 \text{ and } \lambda_{qs} < 0) \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

$$B_3 = \begin{cases} 1, & (\lambda_{qs}/\lambda_{ds} < \tan\pi/3 \text{ and } \lambda_{ds} \geq 0) \\ & \text{or } (\lambda_{qs}/\lambda_{ds} \geq \tan\pi/3 \text{ and } \lambda_{qs} < 0) \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

Equations (18)–(20) show that trigonometric function calculations are not needed for the flux angle encoding, hence, the complexity of implementation is decreased. The flux angle encoder can be accomplished by the hard limit neurons, as shown in Fig. 8.

The five nets are linked together to form a network of 24 different neurons for implementing the flux angle encoder and flux magnitude computation, as shown in Fig. 9.

D. Hysteresis Comparator Sub-Net

Using the hysteresis comparator, as shown in Fig. 10, the flux error between stator flux $|\lambda|$ and its command $|\lambda^*|$ can be limited within $\pm\Delta|\lambda|$, and the flux error code B_6 produced by the hysteresis comparator will be used to select the voltage space vector.

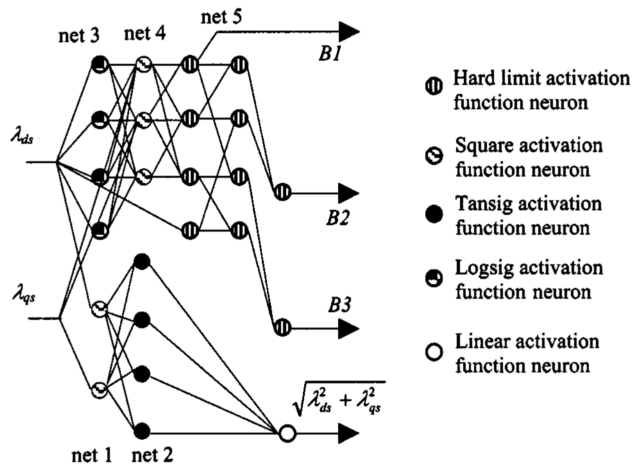

net 3
net 4
net 5
B1
λds
B2
λqs
B3
√(λ²ds + λ²qs)
net 1
net 2
Hard limit activation function neuron
Square activation function neuron
Tansig activation function neuron
Logsig activation function neuron
Linear activation function neuron

Fig. 9. Network of flux angle encoder and flux magnitude computation.

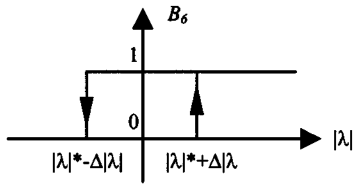


Fig. 10. Flux magnitude hysteresis comparator.

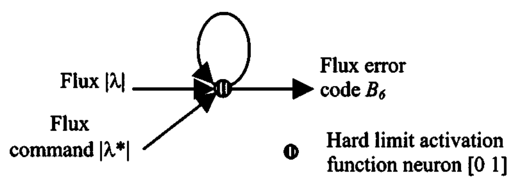


Fig. 11. Flux hysteresis comparator.

The flux error code B_6 can be expressed as

$$B_6 = \begin{cases} 0, & \text{if } |\lambda| < |\lambda|^* + \Delta|\lambda| \text{ and } B_6 = 0 \\ & \text{or if } |\lambda| < |\lambda|^* - \Delta|\lambda| \text{ and } B_6 = 1 \\ 1, & \text{if } |\lambda| \geq |\lambda|^* + \Delta|\lambda| \text{ and } B_6 = 0 \\ & \text{or if } |\lambda| \geq |\lambda|^* - \Delta|\lambda| \text{ and } B_6 = 1. \end{cases} \tag{21}$$

Equation (21) represents a recurrent calculation: *To obtain* B_6*, we have to do a calculation using* B_6*, and in order to do the calculation, we have to obtain* B_6. In order to derive the weight and bias of the network, (21) can be rewritten as

$$B_6 = \begin{cases} 0, & \text{when } (w_1 B_6 + |\lambda| - |\lambda^*| - \Delta|\lambda| < 0) \\ 1, & \text{when } (w_1 B_6 + |\lambda| - |\lambda^*| - \Delta|\lambda| \geq 0) \end{cases} \tag{22}$$

where $w_1 = 2\Delta|\lambda|$.

The output B_6 is connected as an input, which forms a recurrent network [11]. Using the basis function given by (1) for a neural network, the weight and bias can be precomputed. The input of network x is $[B_6,\ |\lambda|,\ |\lambda^*|]$. Its output is B_6, its weight $w = [2\Delta|\lambda| \;\; 1 \;\; -1]$, and its bias $\theta = [-\Delta|\lambda|]$. The flux hysteresis comparator is implemented by a recurrent network with a hard limit function, as shown in Fig. 11.

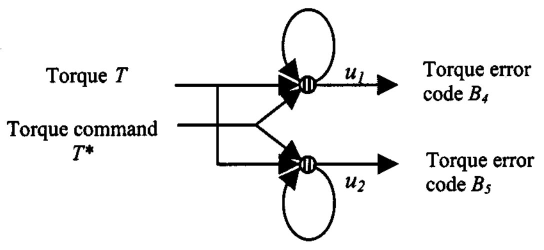
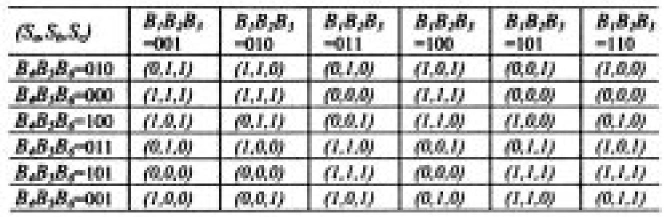


Fig. 12. Torque hysteresis comparator.

TABLE I
DSC OPTIMUM SWITCHING TABLE



(S_a, S_b, S_c)	$B_1B_2B_3$ =001	$B_1B_2B_3$ =010	$B_1B_2B_3$ =011	$B_1B_2B_3$ =100	$B_1B_2B_3$ =101	$B_1B_2B_3$ =110
$B_4B_5B_6$=010	(0,1,1)	(1,1,0)	(0,1,0)	(1,0,1)	(0,0,1)	(1,0,0)
$B_4B_5B_6$=000	(1,1,1)	(1,1,1)	(0,0,0)	(1,1,1)	(0,0,0)	(0,0,0)
$B_4B_5B_6$=100	(1,0,1)	(0,1,1)	(0,0,1)	(1,1,0)	(1,0,0)	(0,1,0)
$B_4B_5B_6$=011	(0,1,0)	(1,0,0)	(1,1,0)	(0,0,1)	(0,1,1)	(1,0,1)
$B_4B_5B_6$=101	(0,0,0)	(0,0,0)	(1,1,1)	(0,0,0)	(1,1,1)	(1,1,1)
$B_4B_5B_6$=001	(1,0,0)	(0,0,1)	(1,0,1)	(0,1,0)	(1,1,0)	(0,1,1)

The difference between the motor torque T and torque command T^* is compared with ΔT and the error flag is used to produce a torque error code for selecting the voltage space vector [5], i.e.,

$$\begin{aligned} &T^* - \Delta T \leq T \leq T^* && (\lambda \text{ rotating in the clockwise direction}) \\ &T^* \leq T \leq T^* + \Delta T && (\lambda \text{ rotating in the counterclockwise direction}). \end{aligned} \tag{23}$$

The torque hysteresis comparator expressed by (23) can be designed in a similar manner as the flux hysteresis comparator. As shown in Fig. 12, the torque error code consists of two bits B_4 and B_5. The weight of u_1 is $[\Delta T \;\; 1 \;\; -1]$ and the bias of u_1 is [0], while the weight of u_2 is $[\Delta T \;\; 1 \;\; -1]$ and the bias of u_2 is $[\Delta T]$.

E. Optimum Switching Table

The DSC optimum switching table is shown in Table I. The flux angle code $B_1B_2B_3$, the torque error code B_4, B_5, and the flux magnitude error code B_6 determine the output voltage codes S_a, S_b, and S_c. The output voltage codes of the optimum switching table represent the on/off status of the inverter switches [5].

A two-layer network with a total of 26 hard limit neurons is employed to implement the optimum switching table as illustrated in Fig. 13. (The first layer has 23 neurons and the second layer has three neurons.) Utilizing the 36 pairs of input and output patterns, shown in Table I, the network is trained by a supervised method with a perceptron training rule [11]. After 321 training epochs, the sum squared error E arrives at zero.

Weights and biases of the trained network of the optimum switching table are shown in the equations at the bottom of the next page.

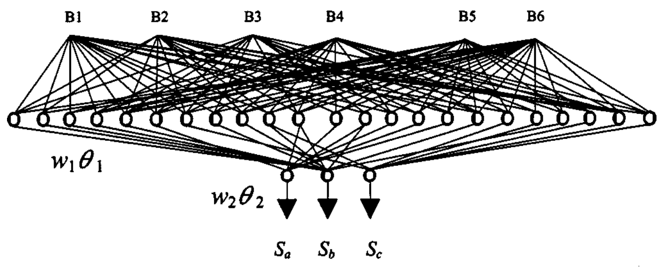


Fig. 13. Optimum switching table implemented by the neural network.

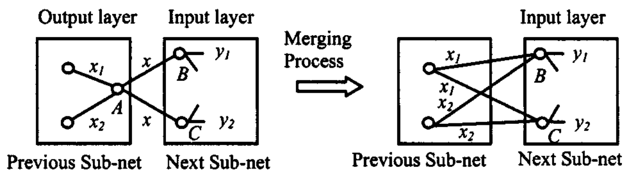


Fig. 14. Merging of neurons with linear activation functions.

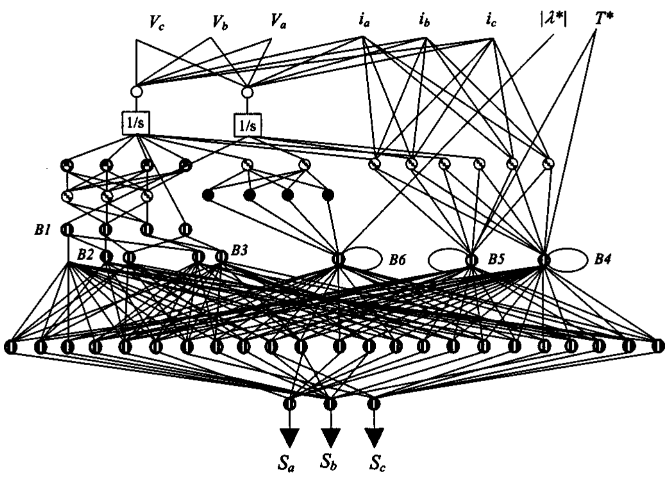


Fig. 15. Neural-network implementation of DSC.

F. Linking of Neural Networks

When the sub-nets are linked to each other, some neurons of the output layer may be merged with the input neurons of the next sub-net. For example, if an output neuron of a sub-net has a linear activation function, it may be merged with the input neuron of the next sub-net. As shown in Fig. 14, the activation function of neuron A is linear, and its output is

$$x = w_1x_1 + w_2x_2 + \theta_1. \tag{24}$$

Let the basis function of neuron B be

$$u_1 = w_3x + w_4x + \theta_2. \tag{25}$$

Substituting (24) into (25),

$$\begin{aligned} u_1 = \; & w_3w_1x_1 + w_3w_2x_2 + w_4w_1x_1 + w_4w_2x_2 \\ & + (w_3 + w_4)\theta_1 + \theta_2. \end{aligned} \tag{26}$$

If the new weight and bias of neuron B are denoted by w' and θ'_2, respectively, then

$$w' = [w_3w_1 + w_4w_1 \;\; w_3w_2 + w_4w_2] \tag{27}$$

$$\theta_2' = \left[(w_3 + w_4)\theta_1 + \theta_2\right]. \tag{28}$$

In this way, neuron A is merged with neuron B, as well as neuron C, as illustrated in Fig. 14.

If an output neuron of a sub-net has a hard limit function, it can also be merged into next sub-net's input neuron that has also a hard limit function. Employing this strategy, the number of layers and neurons of the linked network can be decreased.

With the merging of input and output neurons, the five sub-nets (flux estimation, torque calculation, flux angle encoder and flux magnitude calculation, hysteresis comparator, and optimum switching table) are assembled into the DSC neural network, as shown in Fig. 15.

The complete neural network consists of seven layers and 58 neurons. It may be implemented using special neural devices, such as ASIC chips in the future. Parallelism of neural device

$$w_1 = \begin{bmatrix} 0.8473 & 0.1000 & 0.8214 & 0.9325 & 0.8671 & 0.8807 \\ -0.3959 & 0.6785 & -0.1791 & 0.7114 & 0.1667 & 0.8996 \\ 0.8236 & 0.1031 & -0.2662 & 0.7534 & 0.9052 & -0.9376 \\ 0.9787 & -0.0819 & -0.5239 & -0.7783 & 0.6660 & -0.8659 \\ -0.6582 & 0.0466 & -0.9139 & -0.9580 & 0.5729 & -0.7142 \\ 0.4411 & 0.8341 & -0.2549 & 0.3272 & -0.4276 & -0.5089 \\ -0.6962 & 0.5110 & 0.5568 & 0.5683 & 0.7265 & 0.8997 \\ -0.2369 & -0.5314 & 0.7119 & 0.3871 & -0.8990 & 0.3175 \\ 0.3833 & -0.8297 & -0.9496 & 0.1024 & 0.5905 & 0.7263 \\ 0.4365 & 0.5641 & 0.3388 & 0.6349 & -0.0686 & -0.6487 \\ -0.8817 & 0.5534 & -0.4127 & -0.0749 & -0.1838 & 0.6598 \\ 0.6516 & -0.5826 & -0.3187 & 0.1873 & 0.2835 & -0.6509 \\ -0.1291 & 0.0115 & 0.4511 & 0.6609 & -0.6006 & 0.4883 \\ 0.0408 & -0.5340 & 0.2051 & 0.4697 & -0.4190 & -0.5775 \\ 0.6021 & 0.6986 & 0.9031 & -0.6054 & -0.0458 & 0.6745 \\ 0.9416 & 0.8895 & 0.7088 & 0.4795 & -0.5711 & 0.8700 \\ 0.3989 & -0.5998 & -0.1453 & -0.6091 & -0.8569 & -0.7536 \\ -0.3839 & 0.8020 & -0.6499 & -0.1445 & -0.1848 & 0.4157 \\ 0.7534 & -0.9990 & 0.7505 & -0.5311 & -0.4880 & -0.5296 \\ -0.5086 & 0.6735 & 0.8707 & 0.1676 & -0.1350 & -0.0590 \\ -0.4867 & -0.6002 & 0.7466 & -0.1776 & 0.3418 & -0.0105 \\ 0.7980 & -0.6535 & -0.8298 & -0.3647 & -0.5332 & 0.1036 \\ 0.5562 & 0.0990 & -0.0978 & -0.4616 & -0.1810 & -0.9218 \end{bmatrix} \quad \theta_1 = \begin{bmatrix} 0.3172 \\ -0.8136 \\ 0.2094 \\ -0.3607 \\ 0.2535 \\ -0.0746 \\ 0.1723 \\ -0.7927 \\ -0.5441 \\ -0.7246 \\ -0.8505 \\ 0.1054 \\ -0.0766 \\ 0.2480 \\ -0.0338 \\ -0.9783 \\ -0.7918 \\ -0.6188 \\ -0.2715 \\ -0.2894 \\ -0.8920 \\ -0.4184 \\ 0.6880 \end{bmatrix}$$

$$w_2^T = \begin{bmatrix} 47.0715 & 18.3421 & 39.3128 \\ 19.8300 & 66.1149 & 38.1971 \\ 10.0276 & 10.5172 & -23.2186 \\ 31.3423 & -82.8622 & -2.3431 \\ 81.3558 & -11.4122 & -74.3797 \\ -31.0657 & -15.0781 & -41.1647 \\ 25.5236 & 1.0999 & 34.7033 \\ 50.9470 & -11.9138 & -13.8670 \\ -10.4534 & -31.1892 & 39.6167 \\ 47.5688 & 33.1082 & 67.7114 \\ -32.5416 & -77.4311 & -37.5666 \\ -46.1118 & 9.6021 & -39.3630 \\ -3.9496 & -10.5267 & -15.9416 \\ -57.5613 & -63.8121 & 8.1912 \\ -72.1204 & -16.5997 & 48.0935 \\ -54.5250 & -69.1477 & -103.9177 \\ -0.2557 & -0.6748 & 0.1752 \\ 21.4774 & 8.7560 & 94.9129 \\ 73.1023 & 52.9186 & 25.0732 \\ -2.5922 & -8.4076 & -74.5973 \\ -37.7769 & 34.9370 & -7.5454 \\ 69.0833 & 55.2032 & -45.3538 \\ -23.3127 & 35.2852 & 6.9282 \end{bmatrix} \quad \theta_2 = \begin{bmatrix} 47.0021 \\ 18.0043 \\ 38.4240 \end{bmatrix}$$

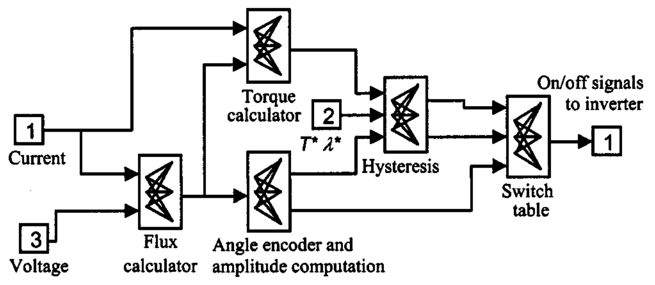


Fig. 16. Neural-network DSC in Simulink window.

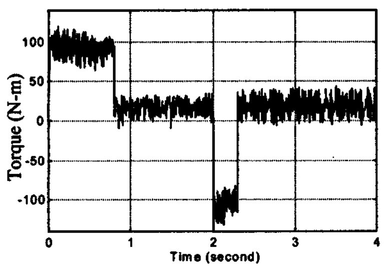


Fig. 17. Torque response of DSP DSC with 100-μs delay.

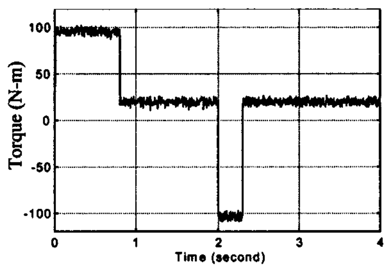


Fig. 18. Torque response of ANN–DSC with 25-μs delay.

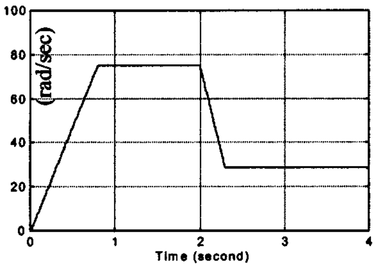


Fig. 19. Speed response of DSP-DSC with 100 μs delay.

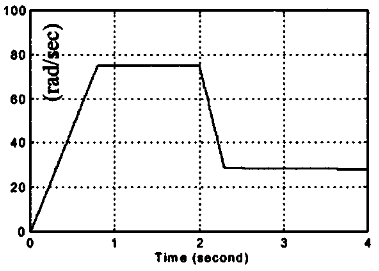


Fig. 20. Speed response of ANN–DSC with 25-μs delay.

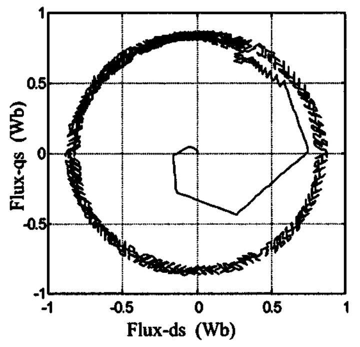


Fig. 21. Flux response of DSP DSC with 100-μs delay (time $= 0$ s$\sim$ 0.2 s).

computation renders the control process extremely fast and the cost should be lower compared with a DSP-based system. In the practical drive, the output potentials of the voltage and current sensors should match the drive voltage of the neural devices, while the output potentials of the neural devices should match gate drive voltage of the inverter.

IV. Simulation of Neural-Network DSC

A Matlab/Simulink program with a Neural Network Toolbox is used to simulate the neural network DSC, as shown in Fig. 16.

The induction motor used for the simulation studies has the following parameters:

Type:	three-phase, 7.5-kW, 220-V, 60-Hz, 6-pole, squirrel-cage induction motor
Stator resistance:	0.288 Ω/ph
Rotor resistance:	0.161 Ω/ph
Stator leakage reactance:	0.512 Ω/ph
Rotor leakage reactance:	0.281 Ω/ph
Magnetizing reactance:	14.821 Ω/ph
Rotor inertia:	0.4 kg $\cdot$ m^2
Inertia of load:	0.4 kg $\cdot$ m^2
Load torque:	20 N $\cdot$ m.

Stator flux and torque commands are as follows:

$$\begin{array}{ll}
|\lambda_s|^* = 0.86 \text{ Wb} & 0 \text{ s} < t \leq 4 \text{ s} \\
T^* = 100 \text{ (N} \cdot \text{m)} & 0 \text{ s} < t \leq 0.8 \text{ s} \\
T^* = 20 \text{ (N} \cdot \text{m)} & 0.8 \text{ s} < t \leq 2 \text{ s} \\
T^* = -100 \text{ (N} \cdot \text{m)} & 2 \text{ s} < t \leq 2.3 \text{ s} \\
T^* = 20 \text{ (N} \cdot \text{m)} & 2.3 \text{ s} < t \leq 4 \text{ s.}
\end{array}$$

Figs. 17, 19, and 21 show the torque response, speed response, and flux response of the classical DSC system with 100-μs controller delay (typical of a DSP-based controller), while Figs. 18, 20, and 22 show the torque response, speed response, and flux response of the neural network control system with 25-μs controller delay. The results demonstrate that DSP-based DSC produces large torque and flux errors, whereas the neural-network controller eliminates almost all these errors. In the simulation studies, it is evident that: 1) torque and flux errors increase with increasing controller delay time and 2) the large torque and flux errors decrease the robustness of the drive system against current noise and load changes. It can be concluded that neural-network-based DSC is a more effective algorithm for the control of an inverter-fed induction motor.

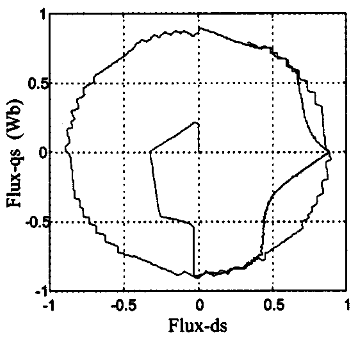


Fig. 22. Flux response of ANN–DSC with 25-μs delay (time $= 0$ s$\sim$ 0.2 s).

V. Conclusion

In this paper, the flexible neural-network structures are used to implement a DSC system for an inverter-fed induction motor drive. Based on the fundamental principle of DSC, an individual training strategy that involves both the fixed weight and supervised networks are employed for the ANN controller design. The fixed-weight method can accurately implement a mathematical model, whereas the supervised method can closely approximate a complex target. Neural-network-based DSC greatly reduces the execution time of the controller, hence, the steady-state control error is almost eliminated. The results of simulation demonstrate that the neural-network algorithm has a better precision in torque and flux responses than the classical DSP-based control method. The application of neural-network techniques simplifies hardware implementation of DSC and it is envisaged that ANN–DSC induction motor drives will gain wider acceptance in the future.

Acknowledgment

The authors would like to thank the Department of Electrical Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong, for providing the computing facilities.

References

[1] M. M. Gupta and N. K. Sinha, *Intelligent Control Systems: Theory and Applications*. Piscataway, NJ: IEEE, 1996.

[2] 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 1993.

[3] K. Rajashekara, A. Kawamura, and K. Matsuse, *Sensorless Control of AC Motor Drives*. Piscataway, NJ: IEEE, 1996.

[4] M. Depenbrock, "Direct self control (DSC) of inverter-fed induction machines," *IEEE Trans. Power Electron.*, vol. 3, pp. 420–429, Oct. 1988.

[5] I. Takahashi and T. Noguchi, "A new quick-response and high-efficiency control strategy of an induction motor," *IEEE Trans. Ind. Applicat.*, vol. IA-22, pp. 820–827, Sept./Oct. 1986.

[6] T. G. Habetler, F. Profumo, M. Pastorelli, and L. M. Tolbert, "Direct torque control of induction machines using space vector modulation," *IEEE Trans. Ind. Applicat.*, vol. 28, pp. 1045–1053, Sept./Oct. 1992.

[7] M. E. Zaghloul, J. L. Meador, and R. W. Newcomb, *Silicon Implementation of Pulse Coded Neural Networks*. Norwell, MA: Kluwer, 1994.

[8] S. Y. Kunk, *Digital Neural Networks*. Engelwood Cliffs, NJ: Prentice-Hall, 1993.

[9] K. L. Shi, T. F. Chan, and Y. K. Wong, "Modeling and simulation of direct self control system," in *Proc. IASTED Int. Conf.*, Pittsburgh, PA, May 1998, pp. ???–???.

[10] A. Delgado, C. Kambhampati, and K. Warwick, "Dynamic recurrent neural network for system identification and control," *Proc. IEE—Control Theory Applicat.*, vol. 142, no. 4, pp. 307–314, 1995.

[11] *Neural Network Toolbox User's Guide*. Natick, MA: The MathWorks, 1994.

K. L. Shi received the B.Sc. degree from the Chengdu University of Science and Technology, Chengdu, China, in 1983, the M.Sc. degree from the Harbin Institute of Technology, Harbin, China, in 1989, and the Ph.D. degree in electrical engineering from The Hong Kong Polytechnic University, Kowloon, Hong Kong.

He was with Honeywell Engineering and Systems, Honeywell Aerospace Canada, Mississauga, ON, Canada. He is currently a Post-Doctoral Fellow in the Electrical and Computer Engineering Department, Ryerson Polytechnic University, Toronto, ON, Canada. His current research interest is the intelligent control and estimation of induction motors.

T. F. Chan (M'95) received the B.Sc.(Eng.) and M.Phil. degrees in electrical engineering from the University of Hong Kong, Hong Kong, in 1974 and 1980, respectively.

Since 1978, he has been with the Department of Electrical Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong, where he is currently an Associate Professor. His current research interests are self-excited ac generators, brushless ac generators, and computer-aided design of electrical machines.

Mr. Chan is a member of the Institution of Electrical Engineers (IEE), U.K. and the Hong Kong Institution of Engineers.

Y. K. Wong (M'80–SM'89) received the B.Sc., M.Sc. and Dip.Ed. degrees from the University of London, London, U.K., in 1977, 1984, and 1987, respectively, the M.Soc.Sc. degree from the University of Hong Kong, Hong Kong, in 1989, the M.Phil. degree from the Chinese University of Hong Kong, Hong Kong, in 1990, and the Ph.D. degree from the Heriot-Watt University, Edinburgh, U.K., in 1997.

In 1980, he joined The Hong Kong Polytechnic University, Kowloon, Hong Kong. He has authored or co-authored over 100 papers in international journals and conference proceedings. His current research interests include linear systems, modeling, simulation, artificial intelligence, intelligent control, and power system control.

Dr. Wong is a Chartered Engineer, Chartered Mathematician, and Chartered Statistician in the U.K. He is a member of the Institution of Electrical Engineers (IEE), U.K., a Fellow of the Institute of Mathematics and Its Applications, and a Fellow of the Royal Statistical Society.

S. L. Ho received the B.Sc. degree (with first-class honors) and the Ph.D. degree in electrical engineering from the University of Warwick Warwick, U.K., in 1976 and 1979, respectively.

In 1979, he joined the Department of Electrical Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong, as an Assistant Lecturer, and then became a Lecturer and Senior Lecturer. He is currently a Full Professor at The Hong Kong Polytechnic University. He has authored or co-authored over 100 papers in international journals and conferences. He is very active in the areas of machine design, phantom loading of motors, condition monitoring of electrical machines, and traction engineering.