A Novel Neural Network Controller and Its Efficient DSP Implementation for Vector-Controlled Induction Motor Drives

Mustafa Mohamadian, Ed Nowicki, *Member, IEEE*, Farhad Ashrafzadeh, *Senior Member, IEEE*, Alfred Chu, Rishi Sachdeva, *Student Member, IEEE*, and Ed Evanik

***Abstract*—An artificial neural network controller is experimentally implemented on the Texas Instruments TMS320C30 digital signal processor (DSP). The controller emulates indirect field-oriented control for an induction motor, generating direct and quadrature current command signals in the stationary frame. In this way, the neural network performs the critical functions of slip estimation and matrix rotation internally. There are five input signals to the neural network controller, namely, a shaft speed signal, the synchronous frame present and delayed values of the quadrature axis stator current, as well as two neural network output signals fed back after a delay of one sample period. The proposed three-layer neural network controller contains only 17 neurons in an attempt to minimize computational requirements of the digital signal processor. This allows DSP resources to be used for other control purposes and system functions. For experimental investigation, a sampling period of 1 ms is employed. Operating at 33.3 MHz (16.7 MIPS), the digital signal processor is able to perform all neural network calculations in a total time of only 280 μs or only 4700 machine instructions. Torque pulsations are initially observed, but are reduced by iterative re-training of the neural network using experimental data. The resulting motor speed step response (for several forward and reverse step commands) quickly tracks the expected response, with negligible error under steady-state conditions.**

***Index Terms*—Field-oriented control, induction motor drives, neural network control, vector control.**

I. Introduction

THE induction motor is the motor of choice in many industrial applications due to its reliability, ruggedness, and relatively low cost. Various methods are now available for the control of induction motor drives, such as the popular constant Volts per Hertz control [1] or the ever more popular field-oriented, or vector, control method [2]–[4]. Recently, ABB has introduced direct torque control (DTC), a speed-sensorless control approach [5]. There has also been some investigation into the application of neural networks to various aspects of induction motor control such as adaptive control [6], sensorless speed control [7]–[10], inverter current regulation [11]–[13], as well as for motor parameter identification purposes [14], [15] and flux estimation purposes [16], [17]. There has been less attention devoted to the implementation of neural-network-based field-oriented control in induction motor drives [18], [19].

In this paper, we propose the use of an artificial neural network to emulate the function of indirect field-orientation control (IFOC). The proposed implementation involves a two stage training process where the second stage employs experimental data. A neural network can be implemented in discreet op-amp form [11] or as a dedicated integrated circuit [20]. More often a personal computer or, as in our case, a digital signal processor (DSP) may be employed. Aside from the research and development versatility that firmware coding permits, a microcontroller, microprocessor, or DSP is often already available in most commercial motor drive systems. Since a digital motor drive system requires a sampling rate on the order of 1 ms, neural network IFOC implementation using the system DSP is a challenging task.

An induction motor is a nonlinear time-varying system, which is difficult to control since state variables are difficult to measure. The variation of rotor resistance resulting from temperature change and saturation of inductance is an issue that must be considered. Nonetheless, an induction motor speed control system is capable of showing high-performance response if it is able to process a large amount of state variable estimation calculations in a small time and also is able to handle the nonlinearities in the system. Neural networks are capable of handling time varying nonlinearities due to their own nonlinear nature. The authors' motivation is to investigate induction motor drive operation with an artificial neural network controller. It is suggested that it is possible and practical to employ a neural network controller that will be insensitive to system parameter variations (such as rotor resistance variation) and take advantage of the learning capability of the neural network. The research discussed below indicates that it is possible to replace a conventional field oriented controller by a neural network controller and achieve the same performance under nominal conditions.

Paper MSDAD-A 03–12, presented at the 2002 Industry Applications Society Annual Meeting, Pittsburgh, PA, October 13–18, and approved for publication in the IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS by the Industrial Automation and Control Committee of the IEEE Industry Applications Society. Manuscript submitted for review October 15, 2002 and released for publication September 18, 2003.

M. Mohamadian is with the Iranian Research Organization for Science and Technology (IROST), Tehran 15815, Iran (e-mail: mohamadian@irost.com).

E. Nowicki and E. Evanik are with the Department of Electrical and Computer Engineering, University of Calgary, Calgary AB T2N 1N4, Canada (e-mail: nowicki@enel.ucalgary.ca; Evanik@enel.ucalgary.ca).

F. Ashrafzadeh is with the R&D Center, Whirlpool Corporation, Benton Harbor, MI 49022 USA (e-mail: Farhad_Ashrafzadeh@Whirlpool.com).

A. Chu is with Telvent Canada Ltd., Calgary, AB T2W 3X6, Canada (Alfred.Chu@telvent.abengoa.com).

R. Sachdeva is with Sustainable Energy Technologies, Calgary, AB T2N 2A1, Canada (e-mail: sachdeva@sustainableenergy.com).

Digital Object Identifier 10.1109/TIA.2003.819441

The objective of this paper is digital signal processor implementation of the neural network controller, obtaining performance at the level possible with a conventional field-oriented controller, while minimizing the computational burden placed on the digital signal processor. This should permit the DSP to still perform system functions. The cost savings resulting from this nonrecurrent engineering effort is especially valuable in high-volume production industries such as those found in the home appliance market.

II. System Description

A. Neural Network Controller

The primary equations that describe the behavior of IFOC are

$$T = K_t i_{qs}^e \lambda_{dr}'^e \tag{1}$$

$$\omega_e = \frac{M}{\tau_r}\frac{i_{qs}^e}{\lambda_{dr}'^e} + \omega_r \tag{2}$$

where the primed notation denotes stator referred. The equations which transform the synchronous reference frame quantities (denoted with an e superscript) to stationary reference frame quantities (denoted with an s superscript) are

$$i_{qs}^s = i_{qs}^e \cos\theta_e + i_{ds}^e \sin\theta_e \tag{3}$$

$$i_{ds}^s = -\, i_{qs}^e \sin\theta_e + i_{ds}^e \cos\theta_e. \tag{4}$$

For (1)–(4),

T	electromagnetic torque;
K_1	motor torque proportionality constant;
i_{qs}^e	synchronous frame q-axis stator current;
i_{ds}^e	synchronous frame $d$axis stator current;
ω_e	rotor flux angular velocity;
θ_e	rotor flux angular position;
ω_r	rotor angular velocity;
$\lambda_{dr}'^e$	stator referred rotor direct axis flux linkage;
i_{qs}^s	stationary frame $q$axis stator current;
i_{ds}^s	stationary frame $d$axis stator current;
τ_r	rotor time constant
M	$3/2\times$ (stator referred stator/motor mutal inductance).

Equations (1)–(4) are often employed in the design of a microprocessor based drive system in order to achieve field-oriented control. In this paper we examine the possible emulation of these equations by a neural network controller.

From (2),

$$\frac{d\theta_e}{dt} = \frac{1}{\tau_r}\frac{i_{qs}^e}{i_{ds}^e} + \omega_r \tag{5}$$

assuming a constant rotor flux magnitude. Using a discrete-time approximation, this differential equation can be transformed into the following difference equation:

$$\left(\frac{1-z^{-1}}{T_s}\right)\theta_e(k) = \frac{1}{\tau_r}\frac{i_{qs}^e(k)}{i_{ds}^e(k)} + \omega_r(k) \tag{6}$$

where T_s is the sampling time, z^{-1} is the delay operator, and k refers to the sampling interval. During each sampling interval the rotor flux angular position is given by

$$\theta_e(k) = T_s\left[\frac{1}{\tau_r}\frac{i_{qs}^e(k)}{i_{ds}^e(k)} + \omega_r(k)\right] + \theta_e(k-1). \tag{7}$$

To calculate $\theta_e(k)$ at $t = kT_s$ information about the following variables is needed: $i_{qs}^e(k), i_{ds}^e(k), \omega_r(k), \theta_e(k-1)$. All of these variables are readily available except $\theta_e(k-1)$. This parameter can also be calculated by (3) and (4) for the previous sampling sample time

$$\theta_e(k-1) = \tan^{-1}\left(\frac{i_{ds}^e(k-1)i_{qs}^s(k-1) - i_{qs}^e(k-1)i_{ds}^s(k-1)}{i_{qs}^e(k-1)i_{qs}^s(k-1) + i_{ds}^e(k-1)i_{ds}^s(k-1)}\right) \tag{8}$$

where $i_{qs}^s(k-1)$ and $i_{ds}^s(k-1)$ can be calculated from (3) and (4).

The system block diagram employing the proposed neural network controller with inputs and outputs chosen based on the above discussion is shown in Fig. 1. It is assumed that the flux command to the controller is constant. The i_{qs}^e signal corresponds to the torque command signal. The neural network controller has three layers, i.e. an input layer, a hidden layer and an output layer. The output layer has two neurons corresponding to the required i_{ds}^s and i_{qs}^s command values, while the hidden layer has five neurons and the input layer has ten neurons. The five inputs to the neural network controller, $\omega_r(k)$, $i_{qs}^e(k)$, $i_{qs}^e(k-1)$, $i_{qs}^s(k-1)$ and $i_{ds}^s(k-1)$, are applied to each of the ten neurons in the input layer. The network is fully connected, i.e., the output of each neuron is connected to all the neurons in the forward layer through a weight. Also, a bias signal is coupled to all the neurons through a weight. All the layers of the neural network have a hyper tangent sigmoid transfer function. There are also normalization and denormalization elements in order to keep the value of the variables in the network between -1 and $+1$. The algorithm used for training is backpropagation [21]. Note that the inner structure of the proposed neural network, i.e., the input layer of ten neurons and one hidden layer of five neurons, is determined by a trial-and-error process where a larger neural network is first employed, and neurons are removed while monitoring system performance. This is for the purpose of reducing the computational requirements of the prototype digital signal processor implementation.

B. Description of the DSP Subsystem

The proposed neural network controller is implemented on a DSP development module, which is installed on an IBM-compatible PC computer.[1] The development module has a Texas Instruments TMS320C30 central processing unit, two analog-to-digital and two digital-to-analog converters and a digital port. There are also some interface circuits for command and data communication between the computer and the development module. Other interface circuits are also designed to interface the development module with a voltage-source inverter, speed sensor, and induction motor current sensors. The

[1]TMS320C30 System Board, Spectrum Digital Inc., Stafford, TX, 1990.

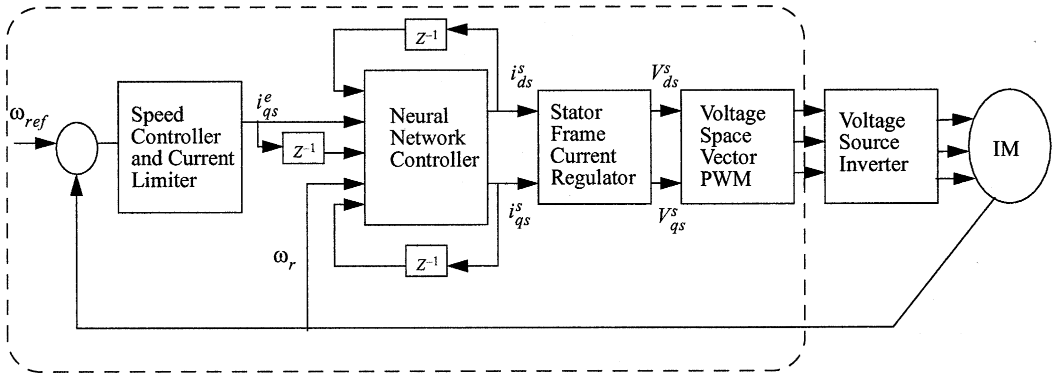


Fig. 1. Block diagram of the proposed speed control drive system with the neural network controller block. The neural network performs as an indirect field-oriented controller, including synchronous frame to stationary frame transformations.

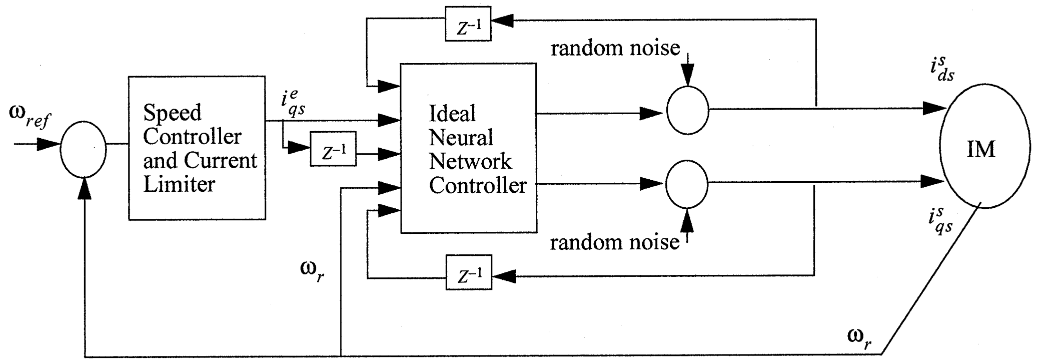


Fig. 2. Block diagram of the speed control drive system with two sources of random noise at the output of the neural network.

TMS320C30 is a 32-bit floating-point processor with 60-ns instruction execution time.

In each sampling period two of the three motor terminal currents are measured by the two analog to digital converters and rotor speed is monitored using the digital port. A stationary reference frame current regulator using voltage space-vector pulsewidth modulation (PWM) is used to regulate the stator current of the induction motor. Device gating signals to the voltage-source inverter are provided through the digital port.

III. Stage One Training

To train the neural network, an induction motor drive system with conventional field-oriented control is first implemented. From the experimental results, neural network input and output training data are obtained. The training data are obtained for several, forward and reverse, speed step commands.

Since the number of neurons in a neural network is limited, there is always an error in the output of the network, even when training data are applied to the neural network. Increasing the number of neurons will decrease this error to a certain limit. Adding more neurons to the network after this limit leads to a problem known as *overfitting*, in which the response of the network to the training data has small error but applying the test data (i.e., data not used for training) shows large error in the output.

The minimum speed step input used as the training data is chosen based on the percentage of speed error for a given error in the neural network output. In order to determine the effects of the neural network output error on induction motor speed, the system of Fig. 2 (which is slightly different from Fig. 1) is simulated. In this figure it is assumed that the neural network is ideal and neural network output currents are the induction motor stator currents, i.e., the inverter transfer function is one. Also, two sources of pseudorandom generated noise are added in the output of the neural network. These two sources simulate the neural network error.

Fig. 3 (simulation results) shows the percentage of output speed error which is produced by an error of 1.5% (arbitrary value) at the output of a neural network for various step speed commands. It is shown in this figure that with an error of 1.5% at the output of the neural network the speed of the induction motor will have a very large percentage of error at lower speeds. Since a source of error with constant maximum amplitude at the output of the ideal neural network will introduce approximately the same amount of absolute speed error at any operating point, the percentage of error at lower speeds as shown in Fig. 3 is higher.

In order to encompass the operating points where the speed error due to the neural network is reasonably low, the training data are chosen to be roughly in the range of 10%–100% of the rated speed (i.e., about 175–1750 r/min for a four-pole machine). For this range, the speed error due to the neural network is less than 30% as shown in Fig. 3.

The induction motor modeled in this paper is a 1/8-hp 230-V four-pole 60-Hz machine. The machine parameters are as follows:

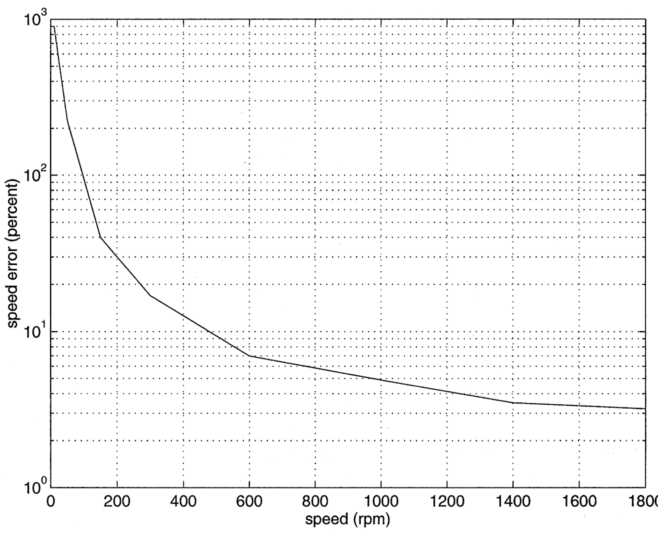


Fig. 3. Output speed error due to a 1.5% error in the neural network outputs (because of the limited number of neurons) simulated for the drive system of Fig. 2 modeling a 1/8-hp four-pole induction motor.

stator resistance	$r_s = 17.2\ \Omega$;
stator referred rotor resistance	$r_r' = 15\ \Omega$;
magnetizing reactance	$X_M = 208.2\ \Omega$;
stator leakage reactance	$X_{ls} = 20.1\ \Omega$;
stator referred rotor leakage reactance	$X_{lr}' = 20.1\ \Omega$;
rotor inertia	$J = 0.0007\ \mathrm{kg\cdot m^2}$.

The training data are produced by applying different speed step inputs to the system. The step inputs are ±200, ±600, ±1000, ±1400, and ±1800 r/min. The training data and the response of the system (c.f. Fig. 1) with the neural network trained after 20 000 iterations (requiring about 30 minutes on a Sun SPARC2 computer) are shown in Fig. 4(a). Also, the normalized error of the neural network output is shown in Fig. 4(b), where the first 5000 data show the error if training data are applied to the neural network and the second 5000 data show the error resulting for the experimental test of the neural network.

As seen in Fig. 4 the results are not satisfactory. A standard procedure in this case to achieve better results is to increase the training data and training iterations. In order to investigate this, the training data for the neural network is increased such that it now includes the previous data plus ±100-, ±400-, ±800-, ±1200-, and ±1600-r/min speed step commands and the neural network is trained for 400 000 iterations (requiring about 10 h on a Sun SPARC2 computer). The response of the system in this case is shown in Fig. 5(a). Also, the error of the output of the neural network is shown in Fig. 5(b). As seen by comparing Figs. 4(b) and 5(b), although the error due to training data is reduced in Fig. 5(b), the experimental results have not improved.

IV. Stage Two Training

The method we chose to improve the performance of the system is to re-train the neural network with input data derived from the experimental test setup of the neural network drive system (Fig. 1). In this stage of training an algorithm is used which is summarized as a flowchart in Fig. 6. This flowchart may be interpreted as follows. The neural network is first trained with ±200-, ±600-, ±1000-, ±1400-, and ±1800-r/min step inputs. Then the experimental output of the system in Fig. 1 with the trained neural network is recorded and inspected. If the results are not within an expected range, the desired output for each set of experimental recorded input data is computed based on the conventional FOC equations, i.e., (1)–(4). The new training data is comprised of this experimental data (which is discarded after each iteration of the algorithm) plus the data from the first stage of training. This process is continued until good speed and torque results are obtained. Results are shown in Fig. 7(a) where the neural network is trained with 10 iterations of the proposed re-training algorithm (400 000 backpropagation iterations). Also, the neural network error for training data and experimental data are shown in Fig. 7(b). The proposed neural network controller now performs as well as its conventional counterpart, under nominal conditions. In Fig. 7(a) the maximum absolute difference between the conventional FOC system response and neural network system is less than 8% of the FOC system response.

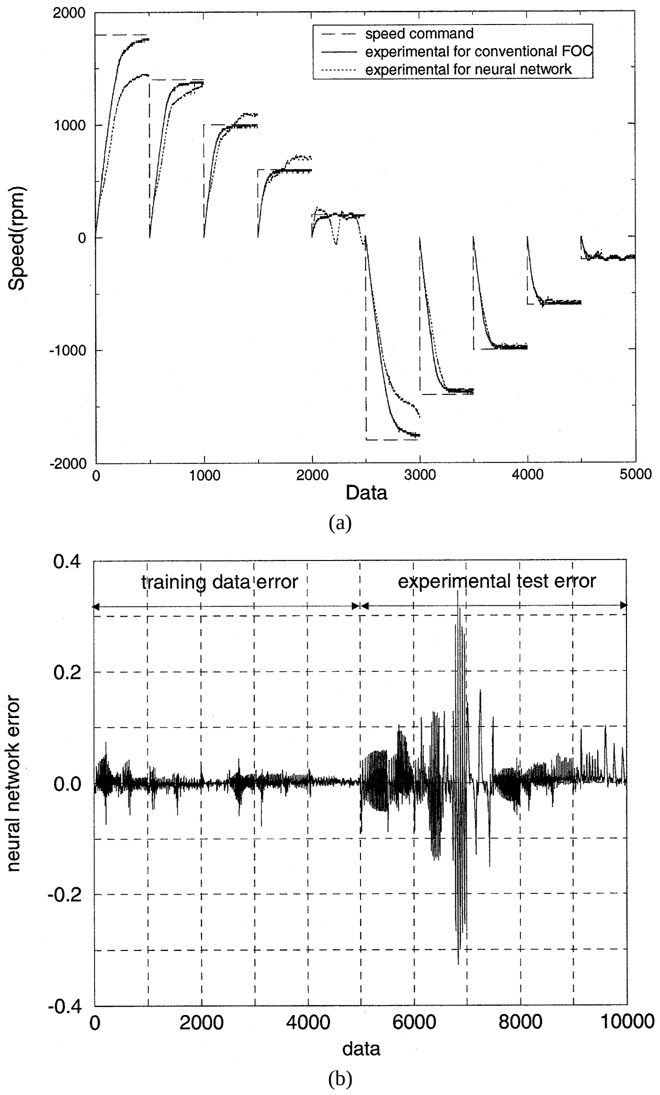
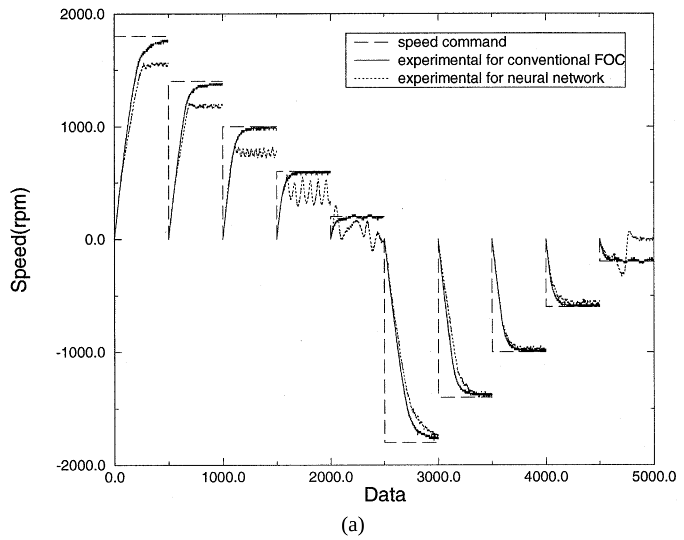


Fig. 4. (a) Experimental training data as produced from a conventional FOC system and experimental data for the neural network speed control system after 20 000 iterations of offline training. (b) Neural network error as found following training and as found experimentally.



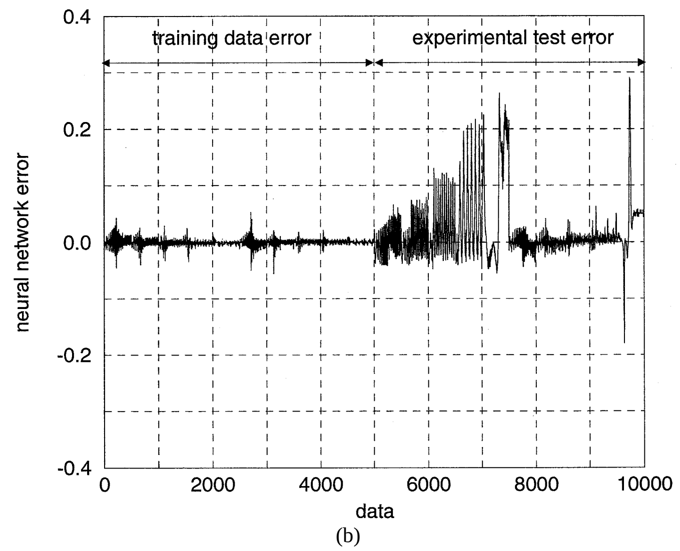


Fig. 5. (a) Experimental training data as produced from a conventional FOC and experimental data for the neural network speed control system after 400 000 iterations of training, i.e., an increase in the number of iterations by a factor of 20 compared to the results shown in Fig. 4. (b) Neural network as found following training and as found experimentally.

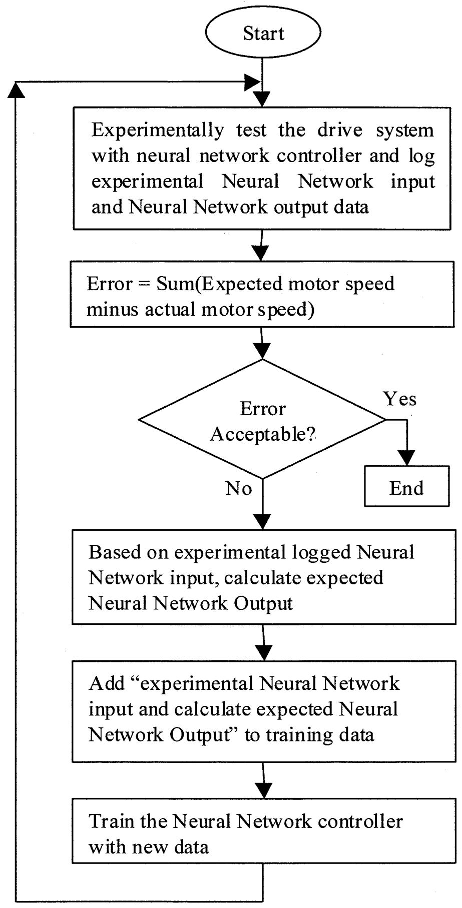

Start
Experimentally test the drive system with neural network controller and log experimental Neural Network input and Neural Network output data
Error = Sum(Expected motor speed minus actual motor speed)
Error Acceptable?
Yes
No
End
Based on experimental logged Neural Network input, calculate expected Neural Network Output
Add "experimental Neural Network input and calculate expected Neural Network Output" to training data
Train the Neural Network controller with new data

Fig. 6. Re-training algorithm for a second stage of neural network training. In this flowchart to produce the training data, the neural network in Fig. 1 is experimentally tested and the desired output based on the conventional FOC equations is calculated for each data set. The training data are updated in each iteration of this flowchart.

V. Assembly Language Coding

Programming structure is important in developing efficient code on a DSP. The solution to this problem is the proper use of the assembly language of the processor. This allows the programmer to efficiently use the instructions and manage memory in order to reduce the computation time. If the programming is performed using a high-level language such as C, different programming styles can affect the neural network computation time.

Referring to Fig. 8, which shows an illustration of one neuron in the neural network, in order to calculate the output of each layer of a neural network, a matrix multiplication must be performed, which may be expressed as

$$out_{num\_lay} = coeff_{num\_lay \times num\_inp} \times in_{num\_inp}$$

where

num_inp	number of inputs;
num_lay	number of neurons in the layer;
$coeff$	weight matrix;
in	input matrix;
out	output matrix.

Programs *multiply_slow* and *multiply_fast* in Appendexes A and B respectively show this multiplication with two different C programs. The difference between these two programs is that all the auxiliary variables in program *multiply_fast* are stored in the DSP internal registers, but in the more straightforward *multiply_slow* program, external memory is used. The other difference is that in the program *multiply_slow*, to access each element of the two dimensional matrix a multiplication and an addition is used, but the *multiply_fast* program takes advantage of the fact that a two dimensional matrix is stored as a one-dimensional matrix in the memory, thus avoiding this calculation.

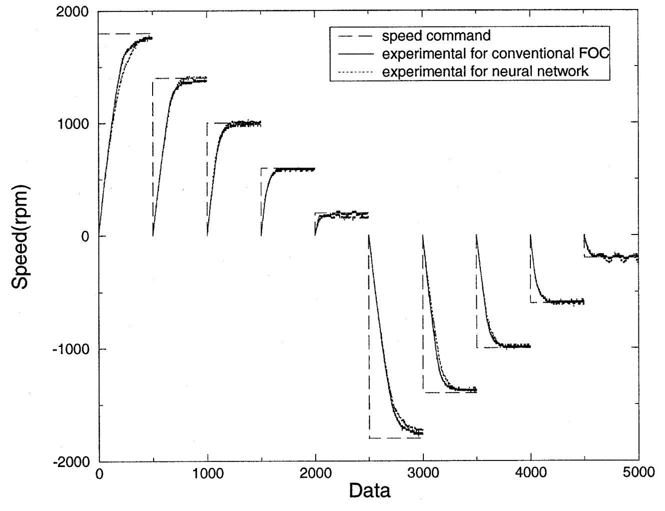
A comparison of the assembly language code of these two programs shows that for the *multiply_slow* program the number of program memory words is 27 and the number of instruction cycles is

$$[(num\_inp \times 17) + 16] \times num\_lay + 2.$$

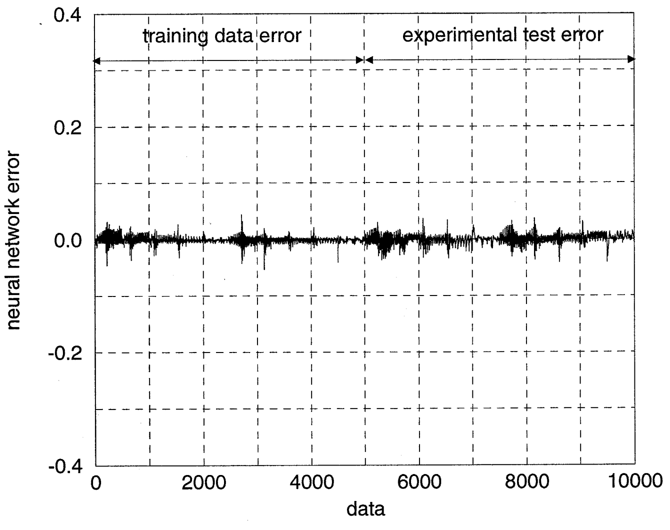
For the *multiply_fast* program, number of program memory words is 24 and number of instruction cycles is

$$[(num\_inp \times 6) + 11] \times num\_lay + 7.$$

For a neural network layer with five inputs per neuron and ten neurons, the *multiply_slow* program required 1012 instruction



(a)



(b)

Fig. 7. (a) Experimental training data as produced from a conventional FOC and experimental data for the neural network speed control system, trained using the algorithm of Fig. 6, after 400 000 iterations of training (ten iterations of the re-training algorithm). (b) Neural network as found following training and as found experimentally. The error is much less compared to Figs. 4(b) and 5(b).

cycles, whereas the *multiply_fast* program requires 417 instruction cycles.

Another coding issue is the calculation of the nonlinearity function F, which is a tangent sigmoid transfer function given by

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{9}$$

The TMS320C30 does not have a tangent sigmoid instruction, but using the C language math library, this function can be implemented using exponential functions. This obvious approach, requires significant processor time, because usually the library functions use the truncated series expansion to calculate the function. A better method to calculate the tangent sigmoid transfer function is to use a lookup table. This method, although introducing a slight error, does not require much processor time.

In order to use the lookup table method, an ordinate interval of -4 to $+4$ is chosen. If the variable x in (9) is less than -4, $f(x)$ is -1 and if x is more than 4, $f(x)$ is $+1$. An array of 32 001

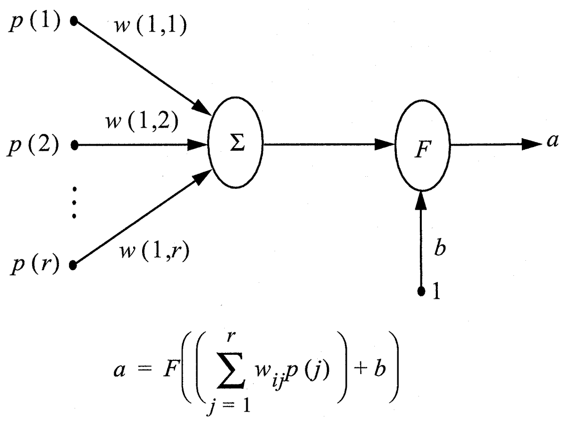

p(1)
w(1,1)
w(1,2)
p(2)
Σ
F
a
p(r)
w(1,r)
b
1
$a = F\left(\left(\sum_{j=1}^{r} w_{ij} p(j)\right) + b\right)$

Fig. 8. A multiple input neuron with bias and nonlinearity function F. In order to implement this neuron on a digital signal processor, r multiplication, r additions, and one calculation of function F are required.

elements is initialized with values of $f(x)$ where x varies from -4 to $+4$. This gives a maximum error of 0.01% in the region of -4 to $+4$ and maximum error of 0.07% outside this region.

Calculations of the proposed neural network controller using the *multiply_fast* routine and the transfer function look up table takes only 280 μs to perform on the TMS320C30 microprocessor operating at 33.3 MHz (16.7 MIPS).

VI. Discussion

In order to reduce the error in the speed tracking of the proposed drive system, one might consider reducing the error at the output of the neural network. There are two sources of output error for the neural network. The first one is a combination of computation error, which is imposed by the 32-bit processing capability of the DSP central processing unit and the approximation in calculating the hyper tangent sigmoid transfer function, which is described in Section V above. In order to determine the magnitude of this combined error we have compared the calculations of the TMS320C30 DSP with the same calculations, which are performed by a computer with 64-bit resolution and using the Borland C^{++} hyperbolic tangent library function [22]. Fig. 9 shows a comparison for the data produced from a 1000-r/min step input. Investigation of this and other operating conditions also indicated that this error is less than 0.11% of the maximum current and rarely goes beyond 0.07%.

The other source of error is the training error. After training, the output of the neural network should *not* be as close as possible to the desired (target) output, as this may lead to overfitting. In this paper, the training error is set to a limit of 4.5%. A comparison of this error with the above-discussed error indicates that the DSP computational error is negligible compared to the training error.

VII. Conclusion

Digital signal processor implementation of a neural network induction motor controller performing field oriented control is presented in this paper. Experimental test results confirm that two stages of training are required to train the neural network, namely, an offline training stage and another stage where experimental data is employed to train the neural network. It is shown

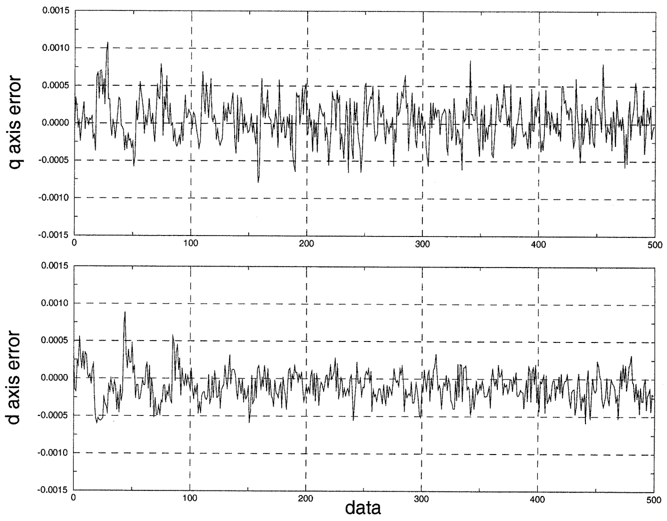


Fig. 9. Comparison of the neural network outputs, implemented on the 32-bit TMS320C30 digital signal processor versus implementation on a computer with 64-bits resolution using Borland C^{++} (DSP calculated current minus computer calculated current). This error is much less than neural network error, c.f. Fig. 7(b). Top graph: error in normalized i_{qs}^s units; bottom graph: error in normalized i_{ds}^s units.

that digital signal processor computation time can be reduced, using simple programming methods, thus enabling the digital signal processor to be used for other system requirements. With the present generation of digital signal processors, the proposed neural network will require only a small portion of processor resources. It is also shown that the neural network error due to computation approximations is negligible compared to the error caused by the limited number of neurons and any lack of information in the training data.

Appendix A
Program to Calculate the Outputs of One Layer of the Neural Network

This program is written in a straightforward style for a high-level programming language. In this program the neural network weight matrix is stored in a two-dimensional matrix and external memory is used to store temporary data.

```
/* multiply_slow.c */
main(){
  long temp1, temp2: /* Temporary vari-
  ables */
  double coeff[NUM_LAY1][NUM_INP];
  /*Coefficient Matrix */
  double bias[NUM_LAY1]; /* Biasmatrix */
  double in[NUM_INP];/* Input matrix*/
  double out[NUM_Lay1]; /*Output Matrix*/
        /* Matrix Multiplication */
  for (itmp1 = 0; itmp1<NUM_LAY1; itmp1++){
    out[itmp1] = 0;
    for (itmp2 = 0; itmp2<NUM_INP; itmp2++){
      out[itmp1]+ = coeff[itmp1][itmp2] * in[itmp2];
    }
    out[itmp1]+ = bias[itmp1];
  }
}
```

Appendix B
Improved C Program to Calculate the Outputs of One Layer of the Neural Network

This program runs faster (more than twice as fast for a neural network layer with ten neurons) on a digital signal processor, compared to the program of Appendix A. In this program the neural network weight matrix is stored in a one-dimensional matrix and DSP internal registers are used to store temporary data.

```
/* multiply_Fast.c */
main(){
  register long itmp1.itmp2; /* Temporary
  Registers */
  register double* npt; /*Pointer to coef-
  ficient matrix*/
  register double* pt1; /*Pointer to input
  matrix*/
  register double* pt2; /*Pointer to
  output matrix*/
  register double* pt3; /*Pointer to bias
  matrix*/
  double coeff[NUM_LAY1*NUM_INP];
          /*Coefficient Matrix*/
  double bias[NUM_LAY1]; /*Bias Matrix*/
  double in[NUM_INP]; /*Input matrix*/
  double out[NUM_Lay1]; /*Output matrix*/
  npt = &coeff[0];
  pt2 = &out[0];
  pt3 = &bias[0];
          /*Matrix Multiplication*/
  for (itmp1 = 0; itmp1<NUM_LAY1; itmp1++){
    *pt2=0
    pt1 = &in[0]
    for (itmp2 = 0; itmp2<NUM_INP; itmp2++){
      (*pt2)+ = (*(npt + +)) * (*(pt1 + +));
    }
    (*pt2)+ = (*(pt3 + +));
    pt2++;
  }
}
```

References

[1] J. Zubek, A. Abbondanti, and C. J. Norby, "Pulsewidth modulated inverter motor drives with improved modulation," *IEEE Trans. Ind. Applicat.*, vol. 11, pp. 695–703, Nov./Dec. 1975.

[2] K. Hasse, "Zur Dynamic Drehzahlgeregelter Antrieebe Mit Stromrichter Gespeisten Asyn-chron Kuzschlublaufermaschinen," Ph.D. dissertation, Technische Hochschule Darmstadt, Darmstadt, Germany, 1969.

[3] F. Blaschke, "Das Verfahren der Feldorientierung zur Regelung der Drehfeldmaschine," Ph.D. dissertation, Univ. Braunschweig, Braunschweig, Germany, 1973.

[4] ——, "The principle of field orientation as applied to the new transvector closed-loop control system for rotating-field machines," *Siemens Rev.*, vol. 34, pp. 217–220, May 1972.

[5] P. Vas, A. F. Stronach, and M. Neuroth, "DSP-controlled intelligent high-performance ac drives present and future," in *IEE Colloq. Vector Control and Direct Torque Control of Induction Motors*, Oct. 1995, pp. 7/1–7/8.

[6] 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, pp. 25–32, Feb. 1995.

[7] D. L. Sobczuk and P. Z. Grabowski, "DSP implmentation of neural network speed estimator for inverter fed induction motor," in *Conf. Rec. IEEE IECON'98*, 1998, pp. 981–985.

[8] P. Vas, A. F. Stronach, and M. Neuroth, "DSP-based speed-sensorless vector controlled induction motor drives using AI-based speed estimator and two current sensors," in *Proc. IEE 7th Int. Conf. Power Electronics and Variable Speed Drives*, 1998, pp. 442–446.

[9] H.-C. Lu, T.-H. Hung, and C.-H. Tsai, "Sensorless vector control of induction motor using artificial neural network," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. II, 2000, pp. 489–492.

[10] Q. Xie, S. Wan, Y. Yi, J. Zhao, and Y. Shen, "Speed-sensorless control using Elman neural network," *J. Syst. Eng. Electron.*, vol. 12, no. 4, pp. 53–58, 2001.

[11] M. R. Buhl and R. D. Lorenz, "Design and implementation of neural networks for digital current regulation of inverter drives," in *Conf. Rec. IEEE-IAS Annu. Meeting*, 1991, pp. 415–423.

[12] D. R. Seidl, D. A. Kaiser, and R. D. Lorenz, "One-step optimal space vector pwm current regulation using a neural network," in *Conf. Rec. IEEE-IAS Annu. Meeting*, 1994, pp. 867–874.

[13] B. Burton, F. Kamran, R. G. Harley, T. G. Habetler, M. A. Brooke, and R. Poddar, "Identifcation and control of induction motor stator currents using fast on-line random training of a neural network," *IEEE Trans. Ind. Applicat.*, vol. 33, pp. 697–704, May/June 1997.

[14] K. Madani, G. Mercier, M. Dinarvand, and J.-C. Dpecker, "A neuro-vector based electrical machines driver combining a neural plant identifier and a conventional vector controller," in *Proc. SPIE 2nd Conf. Applications and Science of Computational Intelligence II*, 1999, pp. 476–485.

[15] L. R. Valdenebro, J. R. Hernandez, and E. Bim, "A neuro-fuzzy based parameter identification of an indirect vector-controlled induction motor drive," in *Proc. IEEE/ASME Int. Conf. Advanced Intelligent Mechatronics*, 1999, pp. 347–352.

[16] A. K. P. Toh, E. P. Nowicki, and F. Ashrafzadeh, "A flux estimator for field oriented control of an induction motor drive," in *Conf. Rec. IEEE-IAS Annu. Meeting*, 1994, pp. 585–592.

[17] M. G. Simoes and B. K. Bose, "Neural network based estimation of feedback signals for a vector controlled induction motor drive," in *Conf. Rec. IEEE-IAS Annu. Meeting*, 1994, pp. 471–479.

[18] A. Barazzouk, A. Cheriti, and G. Olivier, "A neural networks based field oriented control scheme for induction motors," in *Conf Rec IEEE-IAS Annu. Meeting*, 1997, pp. 804–811.

[19] T. Sun and B. Sun, "Research on the application of FNN controller to vector-controlled induction motor drives," in *Proc. Fifth Int. Conf. Electric Machines and Systems*, vol. 2, 2001, pp. 1293–1295.

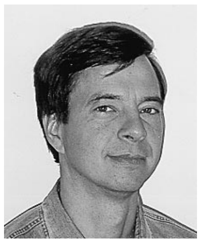
[20] C. Park, K. Buckmann, J. Diamond, U. Santoni, S. The, M. Holler, M. Glier, C. L. Sco_eld, and L. Nunez, "A radial basis function neural network with on-chip learning," in *Proc. IJCNN*, Nagoya, Japan, Oct. 1993, pp. 3035–3038.

[21] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Dept. Appl. Math., Harvard Univ., Cambridge, MA, 1974.

[22] *Borland C++ Version 5.01 User's Guide*, Borland International, Scotts Valley, CA, Mar. 1996.

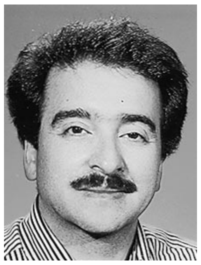
Mustafa Mohamadian received the B.Sc. degree from Amirkabir University of Technology, Tehran, Iran, in 1989, the M.Sc. degree from Tehran University, Tehran, Iran, in 1992, and the Ph.D. degree from the University of Calgary, Calgary, AB, Canada, in 1997, all in electrical engineering.

From 1997 to 2000, he was a Software Engineer with OPISystems Inc., Calgary, AB, Canada, designing real-time microprocessor programs for industrial control and remote data acquisition systems. In 2000, he joined the Iranian Research Organization for Science and Technology (IROST), Tehran, Iran, where he works on electric drives. His current research interests are sensorless electric drives and fixed-point DSP simulation and implementation of control algorithms.



Ed Nowicki (S'87–M'89) received the B.A.Sc. degree in engineering science and the M.A.Sc. and Ph.D. degrees in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 1984, 1987, and 1991 respectively.

He was a Senior Development Engineer with Inverpower Controls Ltd., Burlington, ON, Canada, in the area of medium-voltage induction motor drives, before joining the University of Calgary, Calgary, AB, Canada, in 1992, where he is presently an Associate Professor in the Department of Electrical and Computer Engineering. His current research projects are in the area of switch-mode converter design as applied to alternative energy technologies.



Farhad Ashrafzadeh (S'93–M'97–SM'01) received the B.Sc. degree from Esfahan Institute of Technology, Esfahan, Iran, in 1985, the M.Sc. degree from the University of Science and Technology, Tehran, Iran, in 1988, and the Ph.D. degree from the University of Calgary, Calgary, AB, Canada, in 1996 all in electrical engineering.

He was a Post-Doctoral Fellow at the University of Calgary for about two years. In 1997, he joined the R&D Center, Whirlpool Corporation, Benton Harbor, MI, where he is currently a Lead Engineer, working on advanced control and estimation algorithms for electrical drives. His current interests are system optimization using electrical drives, as well as statistical signal processing for estimation and detection.

Dr. Ashrafzadeh has been the recipient of a number of awards at the Whirlpool R&D Center. He is a certified Six Sigma.

Alfred Chu received the B.Sc. degree in electrical engineering and the M.Sc. degree from the University of Calgary, Calgary, AB, Canada, in 1996 and 1998, respectively.

He has been with Telvent Canada Ltd ., Calgary, AB, Canada, working on SCADA (Supervisory, Control and Data Acquisition) software. His current interests include software design patterns and software project management.

Rishi Sachdeva (S'01) received the B.E. degree in electrical engineering from Thapar Institute of Engineering and Technology, Patiala, India, in 1995, and the M.Sc. degree from the University of Calgary, Calgary, AB, Canada, in 2001.

He was with Crompton Greaves Ltd., Bombay, India, from 1995 to 1997, as a Design Engineer of large induction motors and generators. Upon completing the M.Sc. degree, he worked briefly as a Consultant Engineer with Falcon EDF Ltd., Calgary, AB, Canada, in the area of industrial power systems. Since 2002, he has been with Sustainable Energy Technologies, Calgary, AB, Canada, where he has been involved in developing power electronics products for the renewable energy market. His fields of interests include motor drives, optimal gate drive techniques, and switch-mode converter design.

Ed Evanik received the electronics technology diploma from the Southern Alberta Institute of Technology, Calgary, AB, Canada.

Since 1981, he has been a Technologist in the Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB, Canada, designing and implementing various analog and digital circuits for power electronics research projects as well as for instrumentation and biomedical engineering.