International Journal of Engineering Education Vol. 28, No. 3, pp. 713–723, 2012
Printed in Great Britain

0949-149X/91 $3.00+0.00

A Virtual Laboratory for Neural Network Controlled DC Motors Based on a DC-DC Buck Converter*

OKAN BİNGÖL[1] and SERDAR PAÇACI[2]
[1] Department of Electrical—Electronics-Engineering, Faculty of Technology, SDU, Isparta, Turkey. E-mail: okanbingol@sdu.edu.tr
[2] Department of Information Technologies, SDU, Isparta, Turkey. E-mail: serdarpacaci@sdu.edu.tr

DC-DC converters have a wide usage as the driver circuit of direct current (DC) motors. This has necessitated sensitive speed controls to be made on DC motors. Classical controllers have lower performance due to the non-linear features of DC motors, such as saturation and friction. The Neural Network Controllers (NNC) are widely used in controlling poorly-defined, nonlinear and uncertain systems. NNC courses are now being offered by several universities at the bachelor′s and master's degree levels as a result of NNC's successful applications in these fields. However, the training of an NNC driver circuit in a laboratory environment is a time-consuming and expensive task. In this study, an NNC training set of the DC converter-fed Permanent Magnet Direct Current (PMDC) motor, which is part of the electrical machinery courses, was prepared. The set has a flexible structure and a graphical interface. Thanks to this set, it has become possible to change the PMDC motor and controller parameters, and monitor the system's reaction under various operational conditions in graphics. This training set can also guarantee effective learning and comprehension of Artificial Neural Networks (ANN).

Keywords: DC-DC converter; DC motor; artificial neural network; virtual laboratory

1. Introduction

There has been a rapid increase in computer-based education tools triggered by the developments in information and communication technologies. These training tools have become widely used, particularly for educational purposes in traditional fields of engineering sciences, including electrics, electronics, mechanics, construction, and chemistry. Thanks to their more flexible and interactive nature compared to traditional education methods, computer-based education tools play a more effective role in strengthening the courses taught theoretically, which requires practice in the laboratory environment with computer-based education tools, and in the comprehension of topics by the students. Laboratory environments requiring long periods and high costs are needed in order to be able to acquire practical skills and experiences. Efforts are made to meet this requirement through various lower-cost simulation package software developed for educational purposes, which eliminate the various limitations of traditional experimentation and offer a more convenient educational environment. Package software developed for educational purposes may not be flexible and interactive enough for various sciences.

Therefore, some educators prefer to develop simulation package software with more original graphical user interface and a more flexible structure [1–6].

Recent developments in microprocessors, magnetic materials, semi-conductor technology and mechatronics provided a wide usage for high-performance electrical motors in various industrial processes. Although DC motors are relatively more expensive compared to other electrical motors, they are widely used in many industrial fields, including electrical appliances, electrical trains, cranes, freight elevators, automotive, space and aeronautics, as they are easier to control and demonstrate a higher performance due to their reliability, and because their armature and field currents are decoupled [7, 8]. This has necessitated that speed controls of DC motors be sensitively made. Classical control methods used for DC motor speed controllers prove insufficient in controlling the motor due to the non-linear features of DC motors such as saturation and friction. And thus they reduce the performance of classical controllers [9].

In controlling the DC motor using classical control methods such as Proportional Integral (PI) and Proportional Integral Derivative (PID), the systems need to be very well defined. Appropriate mathematical modeling of the entire system will be needed in order to be able to accurately control a system. However, mathematical modeling of a system with nonlinear structures cannot be fully made. Moreover, many variables of such systems might not be available in a precision allowing for mathematical modeling or such variables may vary by time. During recent years, many adaptive control techniques are proposed for modern driver systems. These are model reference adaptive control (MRAC), sliding mode control (SMC), variable structure

* Accepted 3 January 2012.

control, and self-tuning regulator adaptive control techniques [8]. These classical adaptive control techniques are generally system model parameters-based. Failure to correctly and fully model a system generally leads to problematic design approaches [10].

ANN have learning, generalization and adaptability features. Due to such features, they are widely used in controlling systems with poorly defined mathematical models, and nonlinear and/or uncertain systems [11, 12]. ANN has parallel and distributed information processing structures inspired by the human brain, connected to one another mostly via connections, and each consisting of processing elements with an inherent memory. Because of these features, it is being successfully applied in the business life, finance, industry, education and sciences with complicated problems, in solving problems that cannot be solved using simple methods, and in modeling and controlling nonlinear systems [13]. During recent years, ANN's are being widely used in many fields including prediction [14], estimation [15], function approximation [16], pattern recognition / classification [17, 18], data association [19], data clustering [20], data filtering [21], and optimization [22].

Many universities have included artificial intelligence techniques courses, such as ANNs, fuzzy logic and genetic algorithm, in their associate′s, bachelor′s and master's degree curricula. There are lots of web sites on the internet covering artificial intelligence [23, 24]. MATLAB- Neural Network Toolbox developed by Mathworks Inc., Sharky Neural Network developed by SharkTime Software, NeuroXL developed by Olsoft, Stuttgart Neural Network Simulator (SNNS) developed by the University of Stuttgart, EasyNN-plus developed by Neural Planner Software, NeuroDimension—Neural Network Software developed by NeuroDimension Inc., BrainMaker Neural Network Software developed by California Scientific, SNNAP Neurosimulator Software developed by the University of Texas Medical School at Houston, SIMBRAIN developed by Jeff Yoshimi, PDP++ developed by Randall C. O′Reilly of Carnegie Mellon University etc. are visual package software. However, these software packages are developed with limited capacities, which means they are not suitable for electrical machinery application areas.

In this study, a training set has been developed for a PMDC motor fed by an artificial-neural-network-controlled buck converter. This set consists of the simulations of classical (PI, PID) and artificial-neural-network-controlled models for a speed control of the PMDC motor. The project was developed at the Department of Electronics and Computer Training, Faculty of Technical Education, Süleyman Demirel University. The set may easily be used as training material for students of associate′s, bachelors, and master's degree. It can also assist in developing the curricula of course trainers. The training set was written in C# programming language with Microsoft Visual Studio 2010 IDE using the WPF infrastructure. The set has a flexible structure and a graphical interface. The driver's motor and controller parameters can easily be changed and this allows the monitoring of circuit responses under different operational conditions thanks to the graphics.

2. Mathematical model of the PMDC motor and buck converter

Figure 1 shows a PMDC motor load fed from a buck converter [25]. All elements used in the circuit in the figure are taken to be ideal.

The state equations upon application of Kirchoff's current and voltage laws on the circuit in Fig. 1 are given in Equations (1, 2) in the form of a matrix. Equation (1) shows the relationship of the buck converter.

$$\frac{d}{dt}\begin{bmatrix} i_L \\ V_c \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{L} \\ \frac{1}{C} & 0 \end{bmatrix}\begin{bmatrix} i_L \\ V_c \end{bmatrix} + \begin{bmatrix} \frac{1}{L}S & 0 \\ 0 & -\frac{1}{C} \end{bmatrix}\begin{bmatrix} V_i \\ i_a \end{bmatrix} \tag{1}$$

The armature current and rotor angular speed state equations of the PMDC motor are given in Equations (2) in the form of a matrix. Electromagnetic moment and back electromotor force are given in Equations (3, 4), respectively,

$$\frac{d}{dt}\begin{bmatrix} i_a \\ \omega_r \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{K_b}{L_a} \\ \frac{K_t}{J} & -\frac{B_m}{J} \end{bmatrix}\begin{bmatrix} i_a \\ \omega_r \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} & 0 \\ 0 & -\frac{1}{J} \end{bmatrix}\begin{bmatrix} V_c \\ T_L \end{bmatrix} \tag{2}$$

$$T_e = K_t\, i_a \tag{3}$$

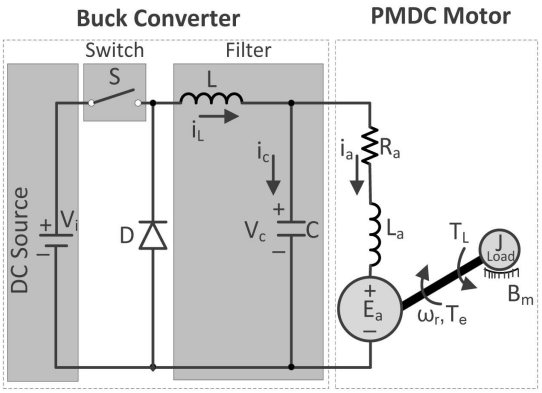

Fig. 1. The PMDC motor and buck converter circuit.

$$E_a = K_b \omega_r \tag{4}$$

where i_L is the coil current (A), V_C is the capacitor voltage (V), L is the converter inductance (H), i_a is the armature current (A), ω_r is the rotor angular speed (rpm), R_a is the armature resistance (Ω), L_a is the armature coil inductance (H), K_b is the back EMF constant (V/(rad/s)), K_t is the moment constant (Nm/A), J is the motor inertia (kg.m^2), B_m is the friction coefficient (Nm/(rad/s), T_L is the load moment (Nm), T_e is the electromagnetic moment (Nm), and E_a is the back electromotor force (back EMF) (V). And the status of the power switch is given as S ∈ [0, 1]. When the switch mode is S = 0, which means it is ON, feed voltage is zero. When the switch mode is S = 1, which means it is OFF, the feed voltage is V_i.

3. The mathematical model and control of the buck converter

Usually, the Pulse Width Modulation (PWM) method is used to control the power switch in DC converters [26]. In this method, the switch's conduction and interruption states are obtained by comparing a DC voltage control signal (V_{cnt}) with voltage signals in the form of repeating saw tooth (V_{swt}). The switch's switching frequency depends on the frequency of the saw tooth signal. Fig. 2 shows the PWM signal. Here, t_{on} is the switch's conduction time, t_{off} is the switch's interruption time and T_s is the switch's sampling time.

4. The artificial neural network controller

ANNs have learning, generalization and adaptability features, and parallel and distributed structures, which allow their speedy operation. They are used in numerous applications thanks to such successful features [13, 27–28]. The first ANN model was developed in 1943 by Warren McCulloch and Walter Pitts [29]. In 1986, Rumelhart and McClelland developed the back-propagation learning algorithm for complicated and multilayer networks [30].

A back-propagation network always consists of at least three layers; an input layer, a hidden layer and an output layer. Fig. 3 shows the topology of ANNs. The weights between the layers as shown in the figure are generally initialized with small random values. This ensures that the networks train and function easier.

The algorithm is simple and relies on propagating an error signal from the output layer backwards towards the input layer through hidden layers. The operation of calculating the output takes place when the input signal is entered into the input layer. The calculation direction is from input layer towards the output layer via hidden layers. The feed-forward computation and the weights adjustments based upon the error are determined during the training. The feedforward computation only takes place in recall [27]. A multilayer feed-forward ANN with one hidden layer is shown in Fig. 3 and an input vector, $X_p = (X_{p1}, X_{p2}, \ldots\ldots X_{pN})$, is applied to the input layer of the network. The error, that is the difference between the desired output (d_{pk}) and the actual output (y_{pk}) of the network in the standard back-propagation algorithm for training of multilayer feed-forward ANN, is based on gradient descent method applied to minimization of total network error (E) [31].

The following equations show the formulation of a standard back-propagation algorithm.

Input values of the hidden layer are shown as follows:

$$net^h_{pj} = \sum_{i=1}^{N} w^h_{ji} x_{pi} + b^h_j \tag{5}$$

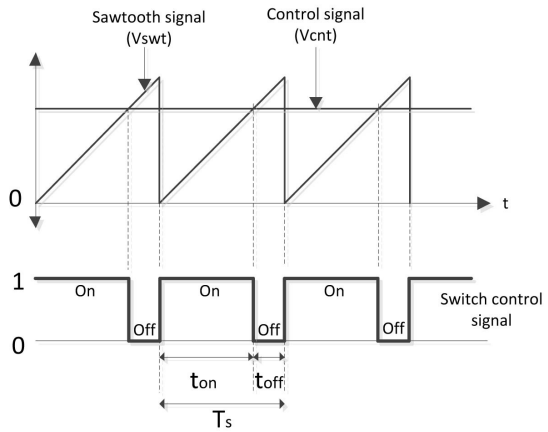

Fig. 2. PWM signal.

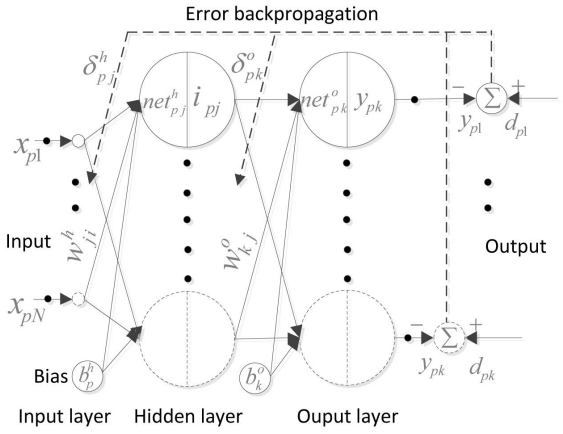

Fig. 3. A multilayer feed-forward ANN with one hidden layer.

Output values of hidden layer are shown as follows:

$$i_{pj} = f_j^h(net_{pj}^h) \tag{6}$$

Input values of output layer are shown as follows:

$$net_{pk}^o = \sum_{j=1}^{L} w_{kj}^o i_{pj} + b_k^o \tag{7}$$

Output values of output layer are shown as follows:

$$y_{pk} = f_k^o(net_{pk}^o) \tag{8}$$

Where f is a transfer function. The transfer function used in training of this study is given as follows:

$$f(net_{pk}^o) = \frac{e^{net_{pk}^o} - e^{-net_{pk}^o}}{e^{net_{pk}^o} + e^{-net_{pk}^o}} \tag{9}$$

The function used in the back-propagation network should be monotonically increasing and continuously differentiable, such as a hyperbolic tangent function, as given in Equation (9). The input and output layers use linear activation functions.

Error of output layer are shown as follows:

$$\delta_{pk}^o = (d_{pk} - y_{pk}) f_k^{o'}(net_{pk}^o) \tag{10}$$

Error of hidden layer are shown as follows:

$$\delta_{pj}^h = f_j^{h'}(net_{pj}^h) \sum_k \delta_{pk}^o w_{kj}^o \tag{11}$$

Network error is shown as follows:

$$E = 1/2 \sum_{k=1}^{M} (d_{pk} - y_{pk})^2 \tag{12}$$

Update weights on the output layer are shown as follows:

$$w_{kj}(t+1) = w_{kj}^o(t) + \eta \delta_{pk}^o i_{pj} + \alpha[w_{kj}^o(t) - w_{kj}^o(t-1)] \tag{13}$$

Update weights on the hidden layer are shown as follows:

$$w_{ji}(t+1) = w_{ji}^h(t) + \eta \delta_{pj}^h x_i + \alpha[w_{pj}^h(t) - w_{pj}^h(t-1)] \tag{14}$$

where η is the learning coefficient, α is the momentum coefficient and t refers to the sampling interval.

5. Speed control of the PMDC motor

The classical PI, PID controller and ANN controllers were applied in order to control the motor speed in the PMDC motor driver system.

5.1 Classical speed control of the PMDC motor

The input variable of classical PI (Proportional + Integral) and PID (Proportional + Integral + Derivative) controllers is defined as the speed error (ω_e) between the reference speed (ω_r*) and the actual speed of the motor (ω_r). And voltage (ΔV_{cnt}) is chosen as the output variable. Fig. 4 shows the block diagram of the PI and PID controllers of the PMDC motor. Equation (15) gives the speed error.

$$\omega_e(k) = \omega_r^*(k) - \omega_r(k) \tag{15}$$

PI and PID controllers are given in Equations (16, 17), respectively. These coefficients are generally named as proportional coefficients (Kp), integral coefficients (Ki) and derivative coefficients (Kd).

$$\Delta V_{cnt} = K_p \omega_e(k) + K_i \sum_{j=0}^{k} \omega_{ej} \tag{16}$$

$$\Delta V_{cnt} = K_p \omega_e(k) + K_i \sum_{j=0}^{k} \omega_{ej} + K_d(\omega_e(k) - \omega_e(k-1)) \tag{17}$$

5.2 NN speed control of the PMDC motor

The input variables of the NN controller are defined as the speed error (ω_e) and the change in speed error (ω_{ce}) between the reference speed (ω_r*) and the motor's actual speed (ω_r). Equation (18) gives the change in speed error. And voltage (ΔV_{cnt}) is chosen as the change in output variable. Fig. 5 gives the block diagram of the neural network controller in the PMDC motor.

$$\omega_{ce}(k) = \omega_e(k) - \omega_e(k-1) \tag{18}$$

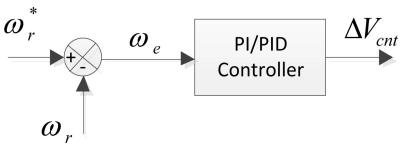

Fig. 4. Classical speed control of the PMDC motor.

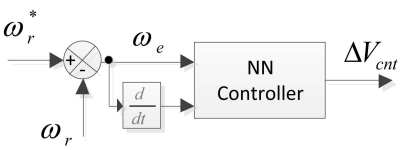

Fig. 5. NN control of the PMDC motor.

Where $\omega_r^*(k)$ is the desired reference speed at the k'th sampling instance, $\omega_r(k)$ is the motor speed at the k'th sampling instance, $\omega_e(k)$ is the motor speed error at the k'th sampling instance, $\omega_{ce}(k)$ is the change in motor speed error at the k'th sampling instance and $\omega_{ce}(k-1)$ is the change in the motor speed error at the $(k-1)$'th sampling instance.

6. Virtual laboratory

The training set has been developed for NN control of PMDC motors, which is written in C# programming language with Microsoft Visual Studio 2010 IDE using the WPF infrastructure and it works in Windows environment. It is developed to help students improve their understanding of NN control of PMDC motors. Using the training set, PMDC motor control performance of the various NN structures can be studied under the parameter and load variations of the motor. This training set is possible to monitor the driver system operations on the computer screen and adjust them choosing the appropriate windows.

The main program's window view is given in Fig. 6. The main window is divided into two parts. These are the control window located on the left of the screen and the menu window located on the right of the screen. The control window and the other window selected can be simultaneously seen by clicking on the desired tab at the top of the screen. No changes can be made to the contents of the control window while the program is under operation. The entire system's operation may be observed in the control window (e.g. PMDC motor operation, simulation time, motor reference speed value, actual speed values, speed error and Buck Converter values). At the upper part of the window, you can find an illustration of the PMDC motor's frontal and side views. The view type selected during the simulation is rotated in the motor's direction of rotation. The diagram of the control system is given in the middle of the control window. In this part, the state of switches as well as the I_L and I_A current, V_C voltage, reference speed, actual speed and speed error values can be seen during the simulation. The controller (PI controller, PID controller and NN controller) may be selected at the bottom part of the controller window. When the NNC is selected as the speed controller, the 'neural network controller setup' window is opened to make the controller settings. You can also set the simulation time at the very bottom of the user control window and control the simulation time using the keys in this part. The 'start' key starts the simulation, the 'pause' key pauses the simulation, the 'stop' key ends the simulation and the 'help' key is used to get help about the set.

The menu window has two sub-windows, with content changing depending on the windows chosen from the menu. When one of the windows is selected, the selected window replaces the other window. The PMDC motor's electrical parameters, buck converter parameters, load parameters and the reference speed value are defined in the PMDC motor and 'buck converter parameters simulation setup' window. For example, a motor may be operated may be operated without any load or under different load conditions depending on the

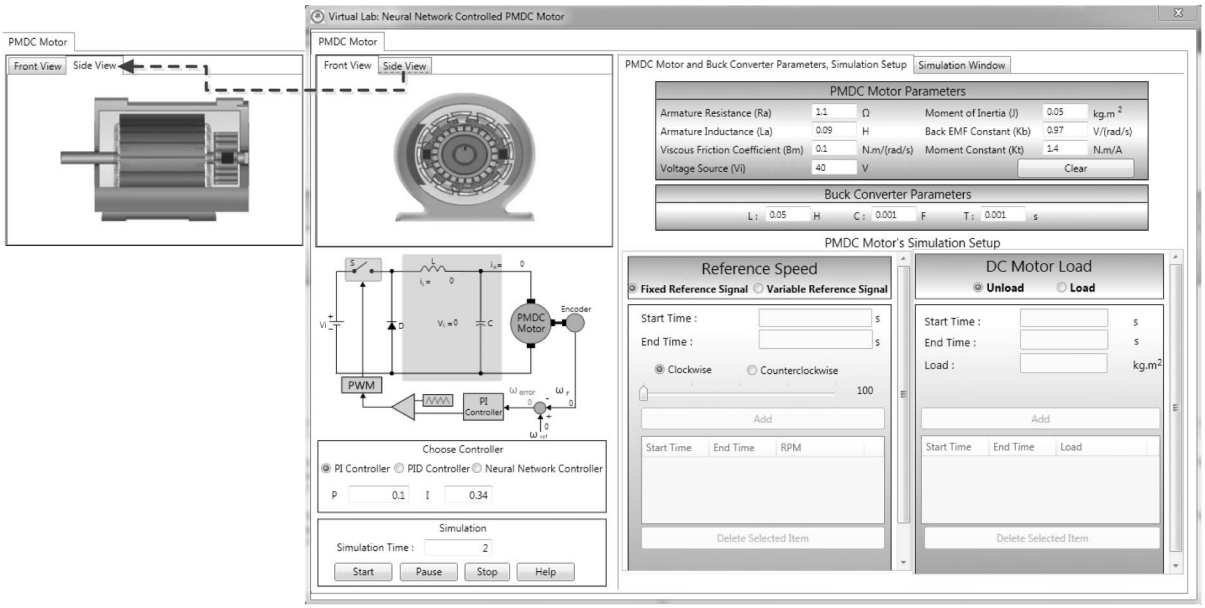

Fig. 6. Main window.

type of load selected. At the same time, the user may define the speed and load function at the desired values in the time interval set throughout the simulation time. In the 'speed setup' part of this window, one of the 'fixed reference signal' or 'variable reference signal' options is selected. When 'fixed reference signal' is selected, the selected value is set as the reference speed throughout the simulation time using the scrollbar. Moreover, the motor's rotation direction is set by selecting one of the 'clockwise' or 'counterclockwise' options. If the 'variable reference signal' is selected, start time and end time values are entered, and again, as in the fixed reference signal, the reference speed is selected using the scrollbar and the 'add' key is pressed. Thanks to this, the motor's reference speed is set in the desired time intervals and desired speed. Previously added reference speed information may be deleted using the 'delete selected item' key.

And the motor's 'unloaded' and 'loaded' operation settings are made in the load setup part. If the loaded option is selected, start time, end time desired load values are entered and the 'add' key is pressed. The 'delete selected item' key is pressed to remove the selected load.

6.1 Neural network controller setup

The main purpose of this module is to model the NNC and set the relevant parameters (weight, threshold value, activation function) in order to carry out the speed control of the PMDC motor. However, the user may use this module completely separately from the motor module, if he/she wishes to. For example, a using intending to use the artificial neural network for a financial analysis can realize the training and testing procedures, use the created ANN model, or may record the same to use it later after installing the relevant training data into the program and modeling its network. The NN setup window is shown in Fig. 7. NNC parameters are defined in this window. On the left part of the NN setup window are the parts showing the data properties, NN properties and training simulation information.

In the 'data properties' part, taking into consideration the data installed or defined by the user itself, the number of outputs to be used in training the network and to be set as reference output values are shown next to the number of inputs set as input values for the ANN. The number of data and the number of used data marked for using in the training are also shown in this part.

In the 'neural network properties' part, the user may see and modify the max iteration number, learning rate constant, momentum constant, mean square error (MSE), number of hidden layers for the modeled network (upon rolling the mouse, all hidden layers defined in the modeled network and the neuron numbers in such hidden layers are shown to the user), number of test data, and number of learning data values. The type of education (online, batch) may also be selected in this part.

At the most bottom of this part, the duration after start of training, the iteration value at t instance during the training and again the MSE value at t instance are shown to the user. In this part, there are

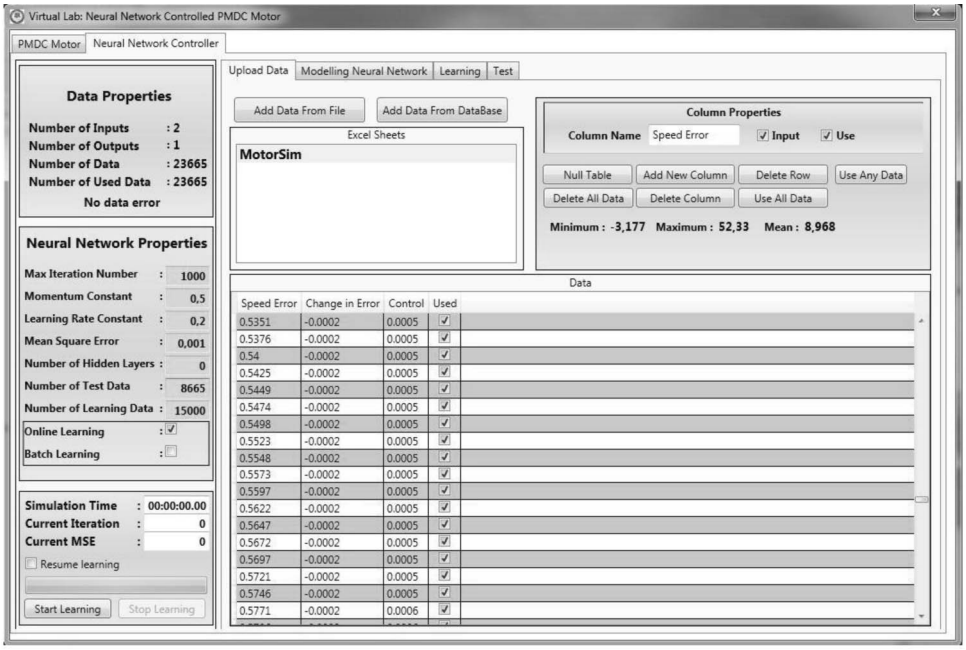

Fig. 7. Neural network controller setup window.

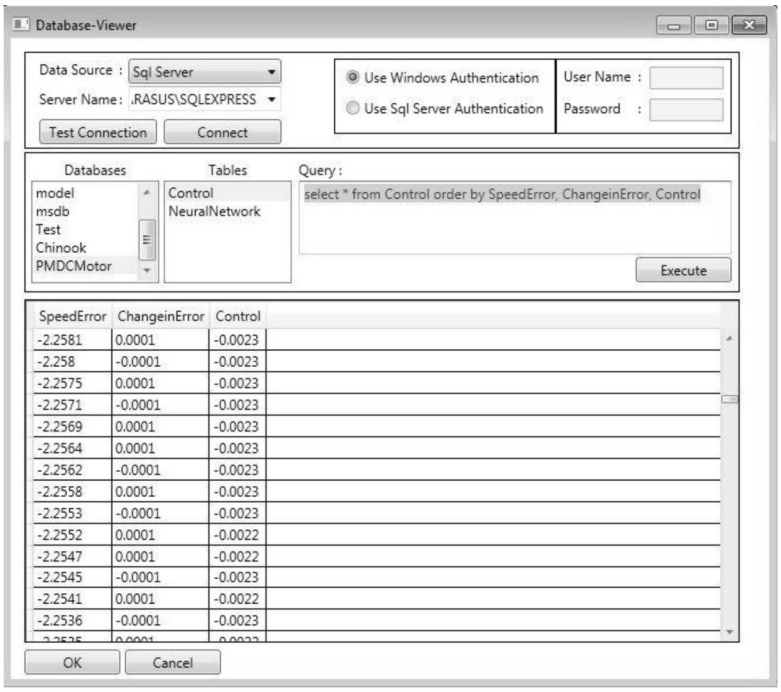

Fig. 8. Screen for adding data from the database.

also the keys that start and pause the learning. If the user wishes to resume with a formerly trained network from where it stopped, he/she marks the resume learning option in this part. Otherwise, the weights and the threshold values will be randomly created and then the network training will start.

NN setup window has four sub-windows, with contents varying depending on the selected window. When one of the windows is selected, the selected window replaces the formed window. These are the upload data, modeling neural network, learning and test windows.

The 'upload data' window allows the user to enter data from a file (excel, access database, csv) or the database (sql server). The user makes changes to add new data on the data he/she entered from a file (add data from file) or database (add data from database) into the software. The user may enter data into the software from a file or a database as well as manually entering the data by creating a blank data table where he/she can create his/her own data using the null table key. Whether or not the entered data can be used in training is set up using the 'used' field in the table. The user may set whether or not the column selected is an input or an output for the ANN and whether or not it will be used in training from the 'column properties' part. Using the keys in this part, the user can add a new column (add new column), delete the selected column (delete column), delete the selected row (delete row), mark all rows as usable (use all data) or mark all rows as non-usable (use any data). Moreover, in this part, the minimum, maximum and arithmetic mean values selected by the user can be seen.

When the user clicks on the 'add data from database' key, the data input window is opened as default, with the sql server being selected. This window is shown in Fig. 8. Any accessible sql server instances are searched in the computer where the user operated the software and over the network if the computer has a network connection, and the results are added into the 'server name' combobox. The user selects a server and authentication mode (windows authentication, sql server authentication) from this list. Using the 'test connection' key, the user tests the selected server connection. The user connects to the database via the 'connect' key. When database connection is established, the database names on the sql server are listed. The tables in that database are listed whenever a database is selected from this list. When the user selects a table, the top 1000 rows in that table are highlighted by default and the relevant sql sentence is sent to the user. From this point onwards, the user may write any sql query and see its results in the table. The 'ok' key transfers the obtained data into the software. And the 'cancel' key cancels the operations made.

The 'modeling neural network' window is the window where the user models its ANN. This window is shown in Fig. 9. The color green shows the input layer, the color blue shows the hidden layers, and the color red shows the output layer in the produced model picture. Here, the user may add

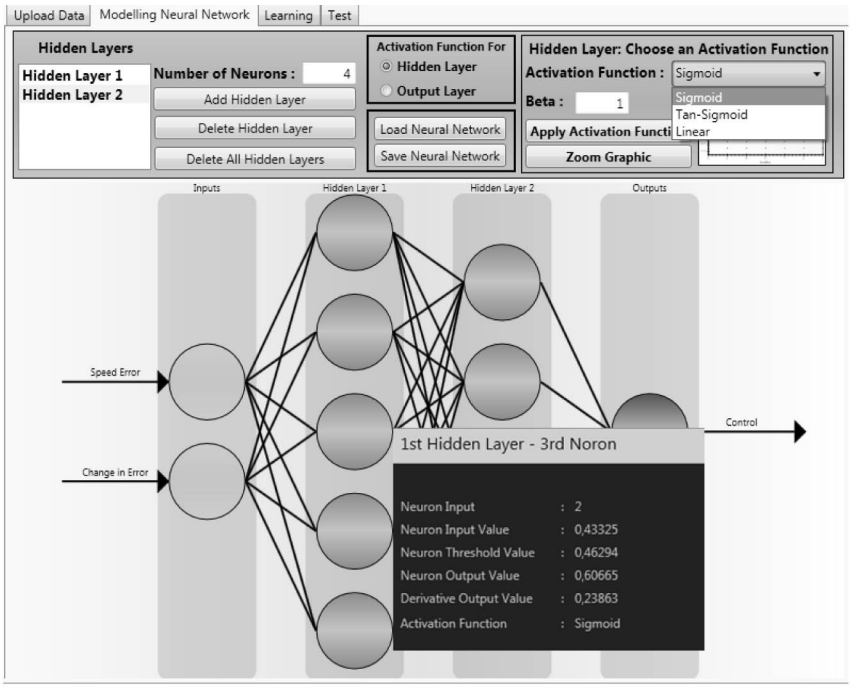

Fig. 9. ANN modeling screen.

as many hidden layers as he wishes using the 'add hidden layer' key and again add any number of neurons as desired on the hidden layers so added. The user may delete the previously added layer using the 'delete hidden layer' key. The 'delete all hidden layers' key is used to delete all hidden layers. When the user adds a layer or specifies the number of neurons in the layer, the network's model is automatically shown to the user. Again, in this part, the user may at any time select an activation function for any desired hidden layer or output layer. The user is given the option to select the sigmoid, tan-sigmoid and linear activation function. When the user selects the activation function and sets the beta value, the graphic of the relevant function can be seen, and if a detailed examination is needed, the graphic can be opened in a separate window using the 'zoom graphic' key. The user can also record the obtained network or install a previously created network into the system. Using the 'save neural network' key, the user may write the weight values between the threshold values of the created artificial neural network cells and the cells into a text file.

And when the 'save neural network' key is pressed, the information of the recorded artificial neural network model is shown to the user. After training the network, when the user rolls over to the neuron using the mouse, information of that neuron (neuron input, neuron input value, neuron threshold value, neuron output value, derivative output value, activation function) are viewed, and when the user rolls over to the weight connections, the weight value of the relevant connection will be viewed.

The learning screen is the screen where the graphics of obtained data are shown. This window is shown in Fig. 10. Using the scroll bar in this window, the user can go to any artificial neural network in the desired iteration and see the data. Using the graphics on the screen, the error information at the output of the artificial neural network, the desired and actual output information in the iterations of data on row basis, the desired output and the network's actual output information by training set and the error information at output may be screened. From the combobox at the bottom of the screen, the data in the desired graphic can be saved to a file (save data to file) or excel (save data to excel).

The testing window is the window where the training results are shown and the artificial neural network is tested. This window is shown in Fig. 11. The table in the upper part of the screen shows the input data for trained data, desired output values, the network's actual output values and the value of output value. And the table at the bottom shows the same information for the test values. The user may select a certain part or all of the training set for training data. The remaining data can be used for testing data. If the user uses the entire data set for training, then the training data set becomes a testing data set as well. In the event that there are few values to be used in training and testing artificial neural network, it may be required that the test values are modified at a selected rate, and the network is tested

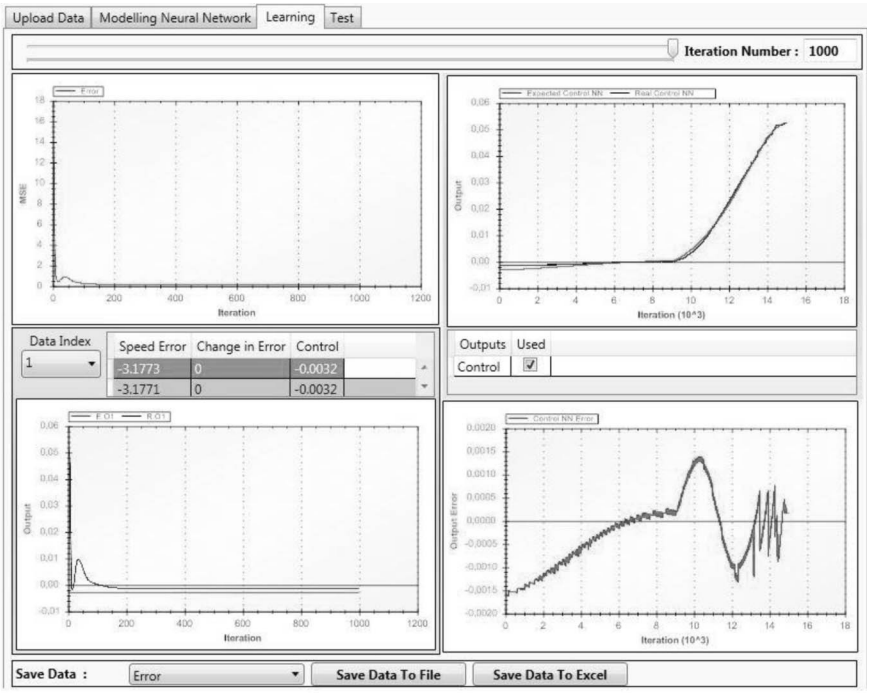

Fig. 10. ANN training screen.

again. In which case, at the bottom of the window, the test input values are modified at a selected rate and the values produced by the network for modified data can be seen using the 'calculate test NN data' key.

The simulation screen is seen in Fig. 12. At the upper part of this screen are given the change graphics by time of PMDC motor speed and the current at the middle section. And the speed error graphic is shown at the bottom part of the screen. By right clicking the mouse on these graphics; Show point values, un-zoom, undo all zoom/pan and set scale to default functions as well as the graphic copy, save as image, and print functions may be selected.

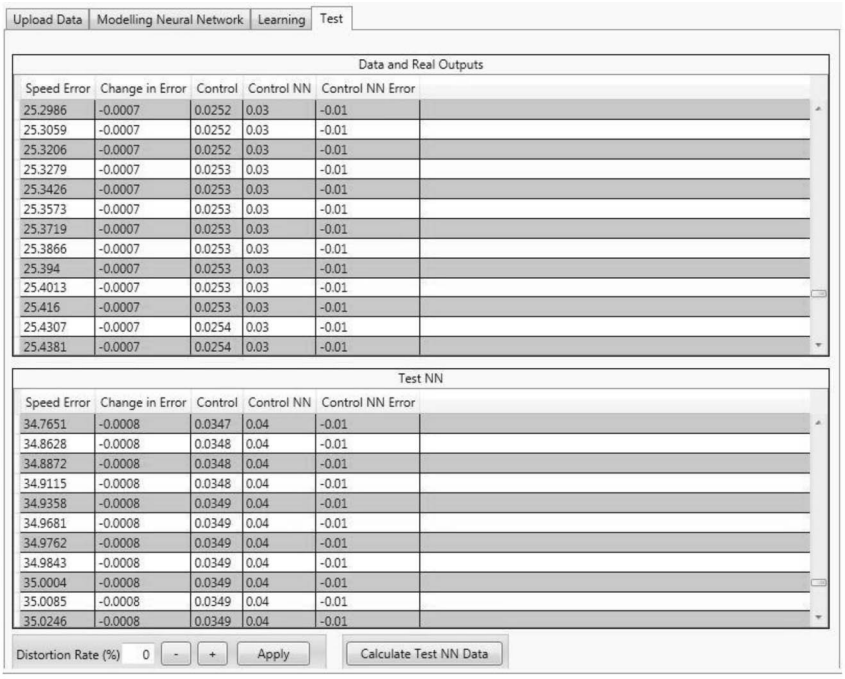

Fig. 11. ANN testing screen.

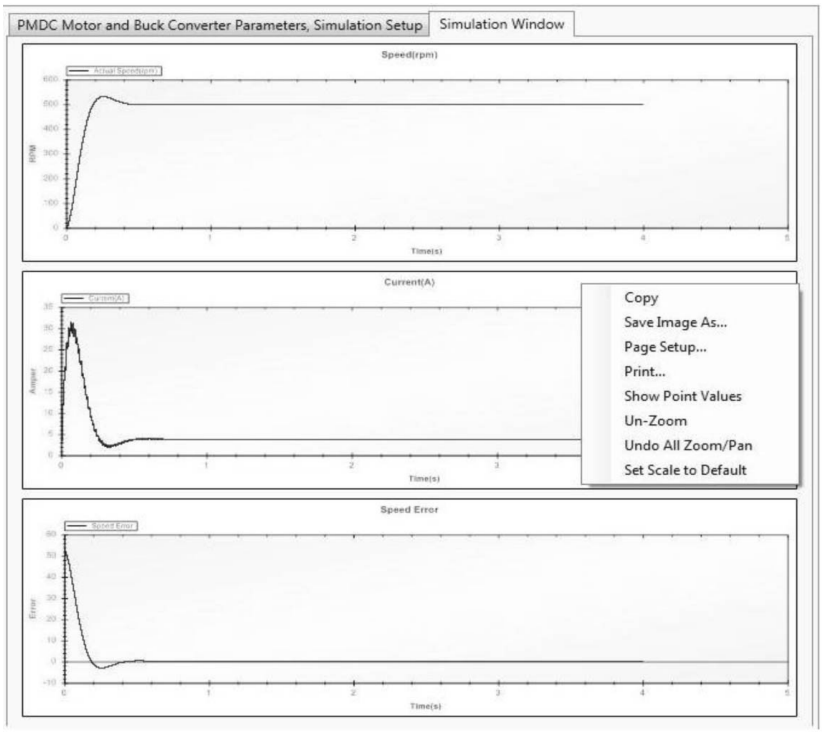

Fig. 12. PMDC motor simulation graphics screen.

7. Implications for teaching and learning

PMDC motor and control is an integrated part of the electrical machinery laboratories practices included in the renewed curricula of the electrics engineering department of universities, departments of electrics education and in vocational high schools. In traditional laboratory experiments, the students do the experiments on their own by forming groups due to the lack of time, resources and sufficient number of lecturer [2, 3]. Generally, there are four or five students in the groups. In larger student groups, the students have no opportunity to perform the most of the experiment. This set aims to overcome the mentioned disadvantages.

Before working with this set, the students need to be theoretically informed about the structure of the PMDC motor, its operation principle, motor and driver modeling as well as classical PI, PID and NN controllers. At the end of the theoretical information acquired during the lesson, the students will be able to easily conduct their relevant experiments with this training, easily strengthening themselves in relevant topics. Therefore, this training set shall allow each student to spend the required time on the set, use different controllers, and learn how the PMDC motor will respond under various speed and load conditions.

The set developed in this study was not tested in an educational environment and its contributions to education was not observed. In future studies, this set should be used in a classroom setting and its effects on educational and motivational conditions should be investigated.

8. Conclusions

This study presents a training set for the PMDC motor driver to ensure cost-efficient training and experience. Classical PI, PID and NN controllers in the PMDC motor driver are used. This training set is useful for the students both in understanding and developing the controllers and the PMDC motor in an effective manner. The set has a flexible structure and a graphical user interface. The user can easily change the driver's motor and controller parameters under various working conditions. The set may easily be installed on a computer operating in the Windows environment (Windows XP, Vista, Windows 7).

References

1. O. Montero-Hernandez, A. Rugerio De La Rosa, D. Baez-Lopez, R. Alejos and E. Enriquez, Power Lab: A Tool to Learn Electrical Machines and Power Electronics, *Computer Applications in Engineering Education*, **7**(4), 1999, pp. 213–220.
2. T. Yigit and Ç. Elmas, An Educational Tool For Controlling SRM, *Computer Applications in Engineering Education*, **16**(4), 2008, pp. 268–279.
3. M. A. Akçayol, Ç. Elmas, O. A. Erdem and M. Kurt, An Educational Tool for Fuzzy Logic Controller and Classical

Controllers, *Computer Applications in Engineering Education*, **12**(2), 2004, pp. 126–135.
4. V. F. Pires and J. F. A. Silva, Teaching Nonlinear Modeling, Simulation, and Control of Electronic Power Converters Using MATLAB/SIMULINK, *IEEE Transactions on Education*, **45**(3), 2002, pp. 253–261.
5. M. Gökbulut, C. Bal and B. Dandil, A Virtual Electrical Drive Control Laboratory: Neuro-Fuzzy Control Of Induction Motors, *Computer Applications in Engineering Education*, **14**(3), 2006, pp. 211–221.
6. E. Tanyldz and A. Orhan, A Virtual Electric Machine Laboratory for Effect of Saturation of the Asynchronous Machine Application, *Computer Applications in Engineering Education*, **17**(4), 2009, pp. 422–428.
7. P. C. Sen, Electric Motor Drives and Control Past, Present and Future, *IEEE Transactions on Industrial Electronics*, **37**(6), 1990, pp. 562–575.
8. M. A. Rahman and M. A. Hoque, On-Line Self-Tuning ANN-Based Speed Control of a PM DC Motor, *IEEE/ASME Transactions on Mechatronics*, **2**(3), 1997, pp. 169–178.
9. C. T. Johnson and R. D. Lorenz, Experimental Identification of Friction and its Compensation in Precise, Position Controlled Mechanisms, *IEEE Transactions on Industry Application*, **28**(6), 1992, pp. 1392–1398.
10. T. Fukuda and T. Shibata, Theory and Applications of Neural Networks for Industrial Control Systems, *IEEE Transactions on Industrial Electronics*, **39**(6), 1992, pp. 472–489.
11. S. Weerasooriya and M. A. El-Sharkawi, Identification and Control of a DC Motor Using Back-Propagation Neural Networks, *IEEE Transactions on Energy Conversion*, **6**(4), 1991, pp. 663–669.
12. K. S. Narendra and K. Parthasarathy, Identification and Control of Dynamical Systems Using Neural Networks, *IEEE Transactions on Neural Networks*, **1**(1), 1990, pp. 4–27.
13. Ç. Elmas, Yapay Zeka Uygulamalar Yapay Sinir Ağlar, Bulank Mantk, Genetik Algoritma. Seçkin Yaynclk, Ankara, 2007, pp. 22–38.
14. C. Alippi and V. Piuri, Experimental Neural Networks for Prediction and Identification, *IEEE Transactions on Instrumentation and Measurement*, **45**(2), 1996, pp. 670–676.
15. X. Q. Liu, H. Y. Zhang, J. Liu and J. Yang, Fault Detection and Diagnosis of Permanent-Magnet DC Motor Based on Parameter Estimation and Neural Network, *IEEE Transactions on Industrial Electronics*, **47**(5), 2000, pp. 1021–1030.
16. W. Y. Wang, T. T. Lee, C. L. Liu and C. H. Wang, Function Approximation Using Fuzzy Neural Networks With Robust Learning Algorithm, *IEEE Transactions on Systems, Man, and Cyberbetics Part B, Cybernetics*, **27**(4), 1997, pp. 740–747.
17. D. Diallo, M. E. H. Benbouzid, D. Hamad and X. Pierre, Fault Detection and Diagnosis in an Induction Machine Drive: A Pattern Recognition Approach Based on Concordia Stator Mean Current Vector, *IEEE Transactions on Energy Conversion*, **20**(3), 2005, pp. 512–519.
18. V. N. Ghate and S. V. Dudul, Cascade Neural-Network-Based Fault Classifier for Three-Phase Induction Motor, *IEEE Transactions on Industrial Electronics*, **58**(5), 2011, pp. 1555–1563.
19. M. A. Zaveri, S. N. Merchant and U. B. Desai, Robust Neural-Network-Based Data Association and Multiple Model-Based Tracking of Multiple Point Targets, IEEE Transactions on Systems, Man, and Cyberbetics Part C, *Cybernetics*, **37**(3), 2007, pp. 337–351.
20. K. L. Du, Clustering: A neural network approach, *Neural Networks*, **23**(1), 2010, pp. 89–107.
21. P. L. Rosin and F. Fierens, The Effects of Data Filtering on Neural Network Learning, *Neurocomputing*, 1998, pp. 155–162.
22. F. J. Lin, L. T. Teng and H. Chu, A Robust Recurrent Wavelet Neural Network Controller With Improved Particle Swarm Optimization for Linear Synchronous Motor Drive, *IEEE Transactions on Power Electronics*, **23**(6), 2008, pp. 3067–3078.
23. O. Bingöl and S. Paçacı, A Virtual Laboratory for Fuzzy Logic Controlled DC Motors, *International Journal of Physical Sciences*, **5**(16), 2010, pp. 2493–2502.
24. M. A. Akçayol, A. Cetin and Ç. Elmas, An Educational Tool for Fuzzy Logic-Controlled BDCM, *IEEE Transactions on Education*, **45**(1), 2002, pp. 33–42.
25. S. E. Lyshevski, Engineering and Scientific Computations Using MATLAB, A John Wiley & Sons Inc., Pub., New Jersey, 2003, pp. 169–170.
26. N. Mohan, T. M. Undeland and W. P. Robbins, Power Electronics: Converters, Applications, and Design JonWiley & Sons, 2 edition, 1995, pp. 162–163.
27. Ç. Elmas, Ş. Sağıroğlu, İ. Çolak and G. Bal, Nonlinear Modelling of a Switched Reluctance Drive Based on Neural Networks, IEEE Melecon 94, 7th Mediterranean Electrotechnical Conference, **2**, 1994, pp. 809–812.
28. Ş. Sağroğlu, İ. Çolak and R. Bayndr, Power Factor Correction Technique Based on Artificial Neural Network, *Energy Conversion and Management*, **47**(18–19), 2006, pp. 3204–3215.
29. W. S. McCulloch and W. Pitts, A Logical Calculus Of The Ideas Immanenet in Nervous Activity, *Bulletin Mathematical Biophysics*, **5**, 1943, pp. 115–137.
30. D. E. Rumelhart and J. L. McClelland, *Paralel Distributed Processing: Explorations in The Microstructure of Cognition*, MIT Press, Cambridge, 1986, pp. 1–1208.
31. J. A. Freeman and D. M. Skapura, *Neural Networks Algorithms, Applications, and Programming Techniques*, Addison-Wesley Publishing Company, 1991, pp. 101–102

Asst. Prof. Dr. Okan Bingöl received his BSc, MSc, and PhD degrees in the Electrical Technologies Education from the Gazi University. He worked as a research assistant at the Gazi University between 1999 and 2005. He worked as a assistant professor at the Suleyman Demirel University between 2005 and 2011. He is currently assistant professor at the Department of Electrical and Electronics Engineering, Faculty of Technology, Suleyman Demirel University, Isparta, Turkey. His research interests are electrical machines, power electronic, educational tools, artificial neural network, fuzzy logic, neuro-fuzzy.

Serdar Paçaci received the BSc and MSc degrees in electronic and computer education from Suleyman Demirel University, Faculty of Technical Education, Isparta, Turkey, in 2007 and 2011, respectively. He is currently pursuing the PhD degree at Sakarya University. He is currently software development specialist at Information Technologies Department of Suleyman Demirel University. His research interests artifical neural network, fuzzy logic, neuro-fuzzy and programming.