Research Article

Design and Implementation of a Novel Intelligent Strategy for the Permanent Magnet Synchronous Motor Emulation

Hamed Zeinoddini-Meymand, Salah Kamel, and Baseem Khan

1Department of Electrical and Computer Engineering, Graduate University of Advanced Technology, Kerman, Iran
2Department of Electrical Engineering, Faculty of Engineering, Aswan University, Aswan 81542, Egypt
3Department of Electrical Engineering, Hawassa University, Awasa, Ethiopia

Correspondence should be addressed to Hamed Zeinoddini-Meymand; h.zeinaddini@gmail.com

Received 17 October 2021; Accepted 11 February 2022; Published 9 March 2022

Academic Editor: Qingling Wang

Copyright © 2022 Hamed Zeinoddini-Meymand et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In this paper, an intelligent neural network-based controller is designed and implemented to control the speed of a permanent magnet synchronous motor (PMSM). First, the exact mathematical model of PMSM is presented, and then, by designing a controller, we apply the wind turbine emulation challenges. The designed controller for the first time is implemented on an Arm Cortex-M microcontroller and tested on a laboratory PMSM. Since online learning neural network on a chip requires a strong processor, high memory, and convergence guarantee, this article uses the offline method. In this method, first, for different work points, the neural network is trained by local controllers, and then, the trained network is implemented on the chip and used. Uncertainty in the parameters and the effect of load torque as challenges of control systems are applied in the proposed method, and a comparison with other methods is performed in the implementation results section.

1. Introduction

In recent decades, fuzzy logic has grown dramatically in many sciences. One of the reasons for the growth of fuzzy logic is that it does not require strong mathematics, and also, in applying fuzzy logic to different systems, a precise dynamic model of the system is not needed [1]. One of the most important reasons for the rapid growth of the use of fuzzy logic is the implementation of human thinking and reasoning as if-then rules. One application of fuzzy logic is fuzzy modeling. Fuzzy modeling involves obtaining “if-then” fuzzy rules from system input-output data, creating the basic model structure (structural identification), and model training (parametric identification) [1]. One way to obtain if-then rules and identify fuzzy model parameters is to combine fuzzy logic and neural networks. This combination can be used as an intelligent controller in various systems. In this paper, the purpose of neuro-fuzzy control is synchronous motor; so first, a brief review of recent work is presented. In [2], the permanent magnet synchronous motor dynamics is first identified by a neural network with postpropagation error training, and then, it is used to adjust the fuzzy control parameters. Using the simulation in this reference, the fuzzy neural network is compared with the PID controller, which shows that the fuzzy neural networks are able to track the reference model with acceptable accuracy as well as the mutation problem of the PID controller. There is no neuro-fuzzy controller. In [3], TSK fuzzy control with reduction gradient training and error propagation method is used to control the six-phase permanent magnet synchronous motor.

Nowadays, combined fuzzy neural methods with sliding mode control [4,5], inverse control [6], robust control [7], H∞ [8], adaptive estimator [9], etc. to control the speed and position of the permanent magnet synchronous motor are widely used. An adaptive neural network distributed control algorithm for a group of high-order non-linear agents with nonidentical unknown control directions under signed time-varying topologies is presented in [10]. An adaptive cooperative control problem with guaranteed convergence
for a class of nonlinear multiagent systems with unknown control directions and time-varying topologies is presented in [11]. In [12], the problem of neural network-based adaptive dynamic surface control for permanent magnet synchronous motors is presented. A decoupling scheme for a bearingless permanent magnet synchronous motor to achieve fast-response is presented in [13].

However, implementing fuzzy neural control hardware with online training is very difficult. Although more than 20 years have passed since the hardware implementation of computational intelligence systems with online training [14], they are still used in the PID industry and fixed structure controllers because they include derivatives and it is an instantaneous update of the parameters, and despite the noise and disturbance in the physical systems, this work has not yet taken place and requires changes in the training system of neural networks and fuzzy neural networks as well as changes in the chips [15]. The combination training method of least-squares and decreasing gradient was introduced in [16]. This method has a high speed in network training and has the least hardware problems and noise cancellation in terms of implementation and has been widely used in recent years [17–22].

In the present paper, the combined least-squares method and descending gradient algorithm and the error propagation method are used to train the adaptive neuro-fuzzy inference system (ANFIS). In the hybrid method, the least-squares method is used to adjust the linear parameters of the output layer, and the descending gradient algorithm is used to adjust the nonlinear parameters of the input layer, including the parameters of the input membership functions. In this paper, a new method based on the fuzzy neural adaptive network is presented offline. In this method, first, the fuzzy neural adaptive network detects changes in motor dynamics at different operating points and with different load torques (with random variable parameters) and is trained by the combined training method of least-squares and decreasing gradients. The identification method is such that for each working point, any amount of turbulence and load torque, as well as random changes of parameters, a precise local controller such as PID, LQR, and LQG is locally designed, and then, the input and output of this controller set is applied to an adaptive fuzzy neural network with combined training and the network covers all controllers. In this case, the trained network has the ability to cope with various situations and the motor speed can be controlled with high accuracy. In the following, first, the dynamic equations of the permanent magnetic synchronous motor will be expressed, and then, the structure of the fuzzy neural adaptive network and its relations will be presented, and finally, the combination of controller and motor, and simulation results will be presented.

2. Dynamic Model of Permanent Magnet Synchronous Motor

The dynamic model of the permanent magnet synchronous motor can be shown as follows [23]:

\[
\begin{align*}
\frac{dI_d}{dt} &= \frac{U_d}{L} - \frac{RI_d}{L} + \omega_r \cdot I_q, \\
\frac{dI_q}{dt} &= \frac{U_q}{L} - \frac{RI_q}{L} - \omega_r \cdot I_d - \frac{\psi_f}{L} \cdot \omega_r, \\
\frac{d\omega_r}{dt} &= \frac{1}{J_m} \left( T_e - T_L - B_m \cdot \omega_r \right), \\
\frac{d\theta_r}{dt} &= \omega_r, \\
\end{align*}
\]

where \( I_d \) and \( U_d \) are the currents and voltages of the \( i_q \), \( i_p \) and \( U_q \) are the currents and voltages of the q-axis, \( \omega_r \) is the rotor angular velocity, \( \theta_r \) is the rotor angular position, \( L \) is inductance, \( R \) is the stator resistance, and \( \psi_f \) is the magnetic flux of the motor. \( T_e \) is the load torque, \( B_m \) is the coefficient of friction, and \( J_m \) is the inertia. The electric torque is obtained from the following equation:

\[
T_e = \frac{3}{2} P \left[ \psi_f \cdot i_q \right],
\]

where \( P \) is the number of motor poles. According to (3), by keeping the current of the d-axis constant, the torque can be controlled with the current of the q-axis. In this regard, (3) can be written as follows:

\[
T_e = K_t \cdot i_q,
\]

where \( K_t \) is a constant value, so by controlling the current \( i_q \), the torque can be controlled.

3. Fuzzy Neural Network

Despite the widespread growth of fuzzy logic, there is still no cinematic process for designing a fuzzy controller [1]. However, a neural network has the ability to see from the training environment (input-output pairs), to arrange its structure and adapt its interaction in some way. Therefore, this problem can be overcome by combining fuzzy logic and neural network. Fuzzy neural networks have the ability to train and extract fuzzy rules and adjust the parameters of membership functions. Figure 1 shows the structure of an adaptive fuzzy neural network with two inputs \( x \) and \( y \) and one output \( f \).

For the structure of Figure 1, two fuzzy rules are defined:

1. If \( x \) is equal to \( A_1 \) and \( y \) is equal to \( B_1 \), then, \( f_1 = p_1 \cdot x + q_1 \cdot y + r_1 \)
2. If \( x \) is equal to \( A_2 \) and \( y \) is equal to \( B_2 \), then, \( f_2 = p_2 \cdot x + q_2 \cdot y + r_2 \)

In Figure 1, the first layer is the fuzzy layer and two membership functions are provided for each of the inputs. The numeric input in this layer is converted to a fuzzy number. For example, if the membership functions are Gaussian, the output of the first layer is calculated as follows:
\[ O_1^i = \mu_{A_k i}(x_i) \]
\[ = \exp \left( -\frac{x_i - m_k i}{\sigma_k i} \right)^2, \quad (5) \]
\[ i = 1, 2, \ldots, n; \quad k = 1, 2, \ldots, l, \]
where \( \mu_{A_k i} \) is the membership function of the linguistic variable \( A_k i \) related to the \( i \)th input and the \( k \)th rule, \( m_k i \) is the center of the Gaussian function and \( \sigma_k i \) is the deviation from the Gaussian function criterion. In the second layer, fuzzy rules are created. If the AND operator is used in the second layer node, the output of the second layer is as follows:
\[ O_2^i = w_i = \mu_{A_k i} \times \mu_{B_k i}, \quad (6) \]
In the third layer, the normalization operation is performed, so that the output of each node from the second layer is divided by the sum of all the outputs of the second layer.
\[ O_3^i = \overline{w_i} = \frac{w_i}{w_1 + w_2}, \quad (7) \]
In the fourth layer, the output of the third layer is multiplied by a function that is a linear combination of inputs.
\[ O_4^i = \overline{w_i} \cdot f_i = \overline{w_i} (p_i x + q_i y + r_i). \quad (8) \]
In a single fifth layer node, the output of all fourth layer nodes is added together.

\[ O_5^i = f = \sum_i \overline{w_i} \cdot f_i, \quad (9) \]

To train the fuzzy neural network, a combination of the least-squares method is used to update the parameters of the fourth layer \( (p_i, q_i, r_i) \), and the error propagation method is used to update the parameters of the Gaussian membership function \( (\sigma_k i, m_k i) \). In the least-squares method, it is assumed that the network output is obtained as follows [6]:
\[ y = \theta_1 \cdot f_1(U) + \theta_2 \cdot f_2(U) + \ldots + \theta_n \cdot f_n(U), \quad (10) \]
where \( U = \left[ u_1, \ldots, u_m \right] \) is the input vector of the network, \( f_1, \ldots, f_n \) are the specific functions of the input vector, and \( \theta_1, \ldots, \theta_n \) are also output parameters that must be updated during network training. It is assumed that for input training, the input-output pair is considered as \( \{(u_i, y_i), \quad i = 1, \ldots, m\} \). (10) can be written as follows:
\[ A\theta = Y, \quad (11) \]
where \( A \) is a matrix \( m \times n \) as follows:
\[ A = \left[ \begin{array}{cccc} f_1(u_1) & \cdots & f_n(u_1) \\ \vdots & \ddots & \vdots \\ f_1(u_m) & \cdots & f_n(u_m) \end{array} \right]. \quad (12) \]
Also, \( \theta \) is the vector parameters of the output as follows:
\[ \theta = \left[ \begin{array}{c} \theta_1 \\ \vdots \\ \theta_n \end{array} \right]. \quad (13) \]
The output is a vector \( m \times 1 \) as follows:
\[
Y = \begin{bmatrix}
y_1 \\
\vdots \\
y_m
\end{bmatrix}
\]

(14)

Since there is always \( m > n \) as well as the modeling error \( \varepsilon \), (11) could be written as follows:

\[
A\theta + \varepsilon = Y.
\]

(15)

The goal of finding \( \tilde{\theta} \) is to minimize the cost function.

\[
E(\theta) = \sum_{i=1}^{m} \left( \left( y_i - a_i^T \cdot \theta \right)^2 \right)
\]

\[
= \varepsilon^T \cdot \varepsilon,
\]

(16)

where \( a_i \) is the \( i^{th} \) row of the matrix \( A \).

If the cost function is minimized, then, \( \theta = \tilde{\theta} \) is obtained.

\[
A^T A \tilde{\theta} = A^T Y.
\]

(17)

If the matrix \( A^T A \) is not singular, \( \tilde{\theta} \) is obtained by solving the following equation.

\[
\tilde{\theta} = (A^T A)^{-1} \cdot A^T Y.
\]

(18)

In equations (10)–(18), if new data is applied to the network, all of the above calculations are repeated. For this reason, it is better to use the least-squares return algorithm. The recursive least-squares algorithm is presented as follows:

\[
P_{k+1} = P_k - \frac{\theta_k \cdot a_{k+1} \cdot a_{k+1}^T \cdot P_k}{1 + a_{k+1}^T \cdot P_k \cdot a_{k+1}},
\]

\[
\theta_{k+1} = \theta_k + P_{k+1} \cdot a_{k+1} \cdot (y_{k+1} - a_{k+1}^T \cdot \theta_k),
\]

(19)

where \( P_k \) is the error variance matrix and \( a_{k+1} \) is the coefficient vector. For example, to estimate the parameters of the output layer of the fuzzy neural network in Figure 1, the values of vectors \( A \) and \( \theta \) are as follows:

\[
f = (w_1 \cdot x_1)^{p_1} + (w_1 \cdot y_1)^{q_1} + w_1 \cdot r_1 + (w_2 \cdot x)^{p_2}
+ (w_2 \cdot y)^{q_2} + w_2 \cdot r_2,
\]

\[
A = \begin{bmatrix} w_1, & xw_1, & yw_1, & w_2, & xw_2, & yw_2 \end{bmatrix},
\]

\[
\theta = [p_1, q_1, r_1, p_2, q_2, r_2]^T.
\]

This section defines the error propagation method with a descending gradient algorithm. It is assumed that the fuzzy neural network has \( L \) layers and the \( f^{th} \) layer has \( N(l) \) neurons. The output neuron \( i \) from layer \( l \) is denoted by the symbol \( x_{iL} \), and the function of neuron \( i \) from layer \( l \) is denoted by the symbol \( f_{iL} \).

\[
x_{iL} = f_{iL}(x_{i-1 \cdot L} , \ldots , x_{i-1 \cdot N(l-1)} , \alpha , \beta , \gamma , \ldots ),
\]

(20)

where \( \alpha , \beta , \gamma , \ldots \) are the parameters of the neurons that must be updated at each stage of training. The total number of data used to train a multinput-multoutput network is assumed to be equal to \( P \). Now if the \( P^{th} \) input-output pair \((1 \leq p \leq P)\) is applied to the network, the sum of the error squares of all outputs related to this data is obtained as follows:

\[
E_p = \sum_{k=1}^{N(L)} \left( d_k - x_{L,k} \right)^2,
\]

(22)

where \( d_k \) is the optimal \( k^{th} \) output of the network and \( x_{L,k} \) is the actual network output of the \( k^{th} \) neuron. Also, \( N(L) \) is the number of the network outputs. In network training with \( P \) data, the goal is to minimize the error associated with all training data, i.e., \( E = \sum_{p=1}^{P} E_p \). This requires minimizing the error of each training data. The basis of the descending gradient algorithm and the error propagation method is based on the derivative, so that in each step the error derivative is calculated to the unknown parameter to obtain the optimal value of the unknown parameter. This starts from the output layer, and the parameters are updated to reach the input layer. The error of the \( i^{th} \) neuron is calculated from the output layer \((L)\) as follows:

\[
\epsilon_{Li} = \frac{\partial E_p}{\partial x_{Li}}
\]

(23)

Similarly, for the \( i^{th} \) neuron from the \( l^{th} \) layer, the error is obtained as follows:

\[
\epsilon_{Li} = \frac{\partial E_p}{\partial x_{Li}}
\]

\[
= \sum_{m=1}^{N(l+1)} \frac{\partial E_p}{\partial x_{Li+m}} \times \frac{\partial f_{iL+m}}{\partial x_{Li}}
\]

(24)

\[
= \sum_{m=1}^{N(l+1)} \epsilon_{Li+m} \times \frac{\partial f_{iL+m}}{\partial x_{Li}}
\]

For example, if \( \alpha \) is the \( n^{th} \) parameter of the \( p^{th} \) layer, the error is obtained as follows:

\[
\frac{\partial E_p}{\partial \alpha} = \frac{\partial E_p}{\partial x_{Li}} \times \frac{\partial f_{iL}}{\partial \alpha} = \epsilon_{Li} \times \frac{\partial f_{iL}}{\partial \alpha}
\]

(25)

\[
\frac{\partial E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha}.
\]

Therefore, the following equation is used to update the parameter during the training:

\[
\alpha_{new} = \alpha_{old} + \Delta \alpha,
\]

(26)

where

\[
\Delta \alpha = -\eta \frac{\partial E}{\partial \alpha}
\]

(27)

To apply the error propagation method, first consider the fuzzy neural network of Figure 2 with two inputs \( \Delta e \) and \( e \) and the output \( I_e^* \), with the Gaussian membership function. In this network, the number of membership functions for the input \( e \) is equal to \( j \) and for the input \( \Delta e \) is equal to \( k \). Network input parameters that are updated
with network training include \( m_{1,j} \) (Gaussian membership function center for input \( e \)), \( m_{2,k} \) (Gaussian membership function center for \( \Delta e \) input), \( \sigma_{1,j} \) (deviation from the Gaussian membership function criterion for input \( e \)), and \( \sigma_{2,k} \) (deviation from the Gaussian membership function criterion for input \( \Delta e \)).

First, the cost function is defined as follows:

\[
E(k) = \frac{1}{2} \left( W_{\text{ref}}(k) - W_a(k) \right)^2
\]

\[= \frac{1}{2} e^2(k), \tag{28}\]

where \( W_{\text{ref}} \) is the reference speed and \( W_a \) is the actual motor speed. The fifth layer error is calculated as follows:

\[
\delta_0^5 = \frac{\partial E}{\partial O_0}\]

\[= e. \tag{29}\]

The fourth layer error is obtained as follows:

\[
\delta_{jk}^4 = -\frac{\partial E}{\partial O_{jk}}\]

\[= \frac{\partial E}{\partial O_{0j}} \frac{\partial O_{0j}}{\partial O_{jk}} \tag{30}\]

The third layer error is obtained as follows:

\[
\delta_{jk}^3 = -\frac{\partial E}{\partial O_{jk}}\]

\[= \frac{\partial E}{\partial O_{0j}} \frac{\partial O_{0j}}{\partial O_{jk}} \frac{\partial O_{jk}}{\partial O_{jk}} \tag{31}\]

The second layer error is calculated for each of the inputs from the following equation:

\[\delta_{jk}^1 = \delta_{jk}^4 \cdot f_{j.k}.\]
\[
\begin{align*}
\delta_{1,j}^2 &= \frac{\partial E}{\partial \delta_{1,j}} = \sum_k \delta_{j,k}^2 \cdot w_{j,k}, \\
\delta_{2,k}^2 &= \frac{\partial E}{\partial \delta_{2,k}} = \sum_j \delta_{j,k}^2 \cdot w_{j,k}.
\end{align*}
\]

In equations (33)-(38), the update of the parameters of the front part and the sequence of fuzzy rules are stated, which includes the update of the parameters of the center and the width of the Gaussian function for each of the inputs [2].

\[
\Delta m_{1,j}(k) = -\frac{\partial E}{\partial m_{1,j}} = -\frac{\partial E}{\partial \mu_{1,j}} \cdot \frac{\partial \mu_{1,j}}{\partial m_{1,j}}
\]

\[
\Delta m_{2,k}(k) = \frac{\partial E}{\partial m_{2,k}} = -\frac{\partial E}{\partial \mu_{2,k}} \cdot \frac{\partial \mu_{2,k}}{\partial m_{2,k}}
\]

\[
\Delta \sigma_{1,j} = \frac{\partial E}{\partial \sigma_{1,j}} = -\frac{\partial E}{\partial \mu_{2,k}} \cdot \frac{\partial \mu_{2,k}}{\partial \sigma_{1,j}}
\]

\[
\Delta \sigma_{2,k} = \frac{\partial E}{\partial \sigma_{2,k}} = -\frac{\partial E}{\partial \mu_{2,k}} \cdot \frac{\partial \mu_{2,k}}{\partial \sigma_{2,k}}
\]

\[
\begin{align*}
m_{1,j}(k) &= m_{1,j}(k-1) + \mu_{m} \cdot \Delta m_{1,j}(k), \\
m_{2,k}(k) &= m_{2,k}(k-1) + \mu_{m} \cdot \Delta m_{2,k}(k), \\
\sigma_{1,j}(k) &= \sigma_{1,j}(k-1) + \mu_{\sigma} \cdot \Delta \sigma_{1,j}(k), \\
\sigma_{2,k}(k) &= \sigma_{2,k}(k-1) + \mu_{\sigma} \cdot \Delta \sigma_{2,k}(k).
\end{align*}
\]

It should be noted that in the combined training method, first the parameters of the first part are considered with initial values and in a fixed form, and then, all the data are applied to the network. Finally, after calculating the parameters of the output layer, the data are applied to the network in a pattern-by-pattern manner and the nonlinear parameters of the fuzzy layer are calculated from the descending gradient algorithm by the error backpropagation method.

4. Use of Fuzzy Neural Network as Speed Controller of the PMSM

The structure of the drive used with the fuzzy neural controller to control the motor speed is shown in Figure 3. Fuzzy neural network is first trained offline by combining two methods of least-squares and descending gradient with the error propagation method and after training the network and placing it as a controller in the block diagram of Figure 3, the motor speed can be controlled. The input signals of the controller are \(e\) and \(\Delta e\), and the output signal of the controller is \(I_d^*\).

In Figure 3, the control inputs are obtained from the following equations:

\[
e(k) = W_{ref} (k) - W_{q} (k), \quad c(k) = e(k) - e(k - 1).
\]

In Figure 3, the two PI controllers act as a flow control loop. If the \(i_d\) current is controlled by a zero reference current, the PMSM can be controlled and controlled by the \(i_q\) current like a DC motor [24]. Figure 3 uses IGBT modules in the inverter.

5. Hardware Implementation

The structure of Figure 3 has been used to evaluate the proposed method. Figure 4 shows the ARM Cortex-M board, and also, the laboratory setup of the PMSM control system is shown in Figure 5.

Specifications of the 4-pole permanent magnet synchronous motor used are described in Table 1 [24].

In the used fuzzy neural network, five membership functions are designed for each input. The membership functions used are of the Gaussian type. In Figures 6 and 7, the membership functions of each of the inputs are shown.

In Figure 6, the input space \(e\) is divided into five parts: very small, small, medium, large, and very large. Also, in Figure 7, the input space \(c\) is divided into five parts: large negative, medium negative, small negative, negative, and zero. Table 2 shows 25 fuzzy rules. As mentioned earlier, the output of the first-order Sugeno neural-fuzzy network is a linear combination of inputs and is a nonfuzzy number. In Table 2, the linear parameters of the control signal \(I_d^*\), i.e., \(p\), \(q\), and \(r\) are expressed for each rule, respectively.

Figure 8 shows the speed of a permanent magnet synchronous motor using a neural-fuzzy controller.

Figure 9 shows the performance of three types of controllers ANFIS, Fuzzy, and PID.

In Figure 10, in order to show the ability of the proposed structure to deal with uncertainty, the motor parameters \(R\) and \(L\) are halved.
Figures 11–13 show the motor torque \((T_e)\) for normal mode, halving the parameters and doubling the parameters, respectively.

Figure 14 shows the performance of the three controllers ANFIS, Fuzzy, and PID for reference speed as a “step.”

Figure 15 is a magnification of the transient part of Figure 14.

Figure 16 shows the q-axis current of all three ANFIS, Fuzzy, and PID controllers in the “step” reference input mode.

Figure 17 shows the d-axis current of all three controllers ANFIS, Fuzzy, and PID, in the “step” reference input mode. As can be seen, the current on the d-axis is in the mA range.

Figure 18 shows the PWM signal of all three ANFIS controllers, Fuzzy, and PID, in the “step” reference input mode.

In Figure 19, part of Figure 18 is magnified. In the following, our proposed method is compared with the method of [15]. In Figure 20, the PMSM speed is compared exactly with the method of [15] with its frame.

In order to better compare our proposed method in this paper with the reference method [15], it is assumed that at the moment \(t = 125\) s, it is applied to the motor once as \(T_1 = 2\) Nm, and then, at the moment \(t = 175\) s, it is separated from the motor. Figure 21 shows the state of motor speed following the “step” reference speed (applied from zero to 500 rpm at \(t = 50\) s) with the above load conditions.

In Figure 22, Figure 21 is zoomed at \(t = 125\) s.

In Figure 23, Figure 21 is zoomed at \(t = 175\) s.

As expected and shown in Figures 21 to 23, the response speed of our proposed method is much faster than the reference method [15]. In general, it can be stated that in Figure 6, the neural-fuzzy controller has been able to follow

---

Table 1: Parameters and nominal values of the PMSM.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbols</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal power</td>
<td>(P_n)</td>
<td>2 kW</td>
</tr>
<tr>
<td>Rated voltage</td>
<td>(V_{L-L})</td>
<td>380 V</td>
</tr>
<tr>
<td>Nominal stream</td>
<td>(I)</td>
<td>5 A</td>
</tr>
<tr>
<td>Nominal speed</td>
<td>(\omega_r)</td>
<td>380 rad/s</td>
</tr>
<tr>
<td>Nominal torque</td>
<td>(T_e)</td>
<td>3.5 Nm</td>
</tr>
<tr>
<td>Stator resistance</td>
<td>(R_s)</td>
<td>1.6 (\Omega)</td>
</tr>
<tr>
<td>Induction</td>
<td>(L_s)</td>
<td>0.5 mH</td>
</tr>
<tr>
<td>Magnetic flux</td>
<td>(\phi_f)</td>
<td>0.12 Wb</td>
</tr>
<tr>
<td>Inertia</td>
<td>(J_m)</td>
<td>0.000552 Nm/(s^2)</td>
</tr>
<tr>
<td>Friction coefficient</td>
<td>(B_m)</td>
<td>0.00035 Nms/rad</td>
</tr>
</tbody>
</table>
the reference speed well. It should be noted that in order to train the fuzzy neural network, the input-output information of Fuzzy and PID controllers has been used. So, to put it simply, the adaptive fuzzy neural network uses the Fuzzy and PID controllers in its heart. As can be seen, the performance of the proposed method in the adaptive fuzzy neural network is better than fuzzy alone. Figures 8 and 9 showed the proper performance of the proposed method by considering the uncertainty in the parameters of the permanent magnet synchronous motor. In Figure 10, a load torque of one Nm was applied at the moment $t = 100$ s, which again proved the proper performance of our proposed method. In Figures 11–13, the motor torque was depicted, and finally, in Figure 14, the performance of the controllers was measured.

![Figure 6: Membership functions for input $e$.](image)

![Figure 7: Membership functions for input $c$.](image)

<table>
<thead>
<tr>
<th>$e$</th>
<th>$c$</th>
<th>VS</th>
<th>S</th>
<th>M</th>
<th>B</th>
<th>VB</th>
</tr>
</thead>
<tbody>
<tr>
<td>$4 \times 10^{-5}$</td>
<td>$-0.022$</td>
<td>0.375</td>
<td>0.0019</td>
<td>0.0066</td>
<td>0.0019</td>
<td>0.0066</td>
</tr>
<tr>
<td>$-8 \times 10^{-6}$</td>
<td>$-8 \times 10^{-4}$</td>
<td>$-0.021$</td>
<td>0.019</td>
<td>0.0277</td>
<td>0.019</td>
<td>0.0277</td>
</tr>
<tr>
<td>$2 \times 10^{-7}$</td>
<td>$2 \times 10^{-7}$</td>
<td>0.019</td>
<td>0.0515</td>
<td>0.0019</td>
<td>0.0515</td>
<td>0.0019</td>
</tr>
<tr>
<td>$-0.001$</td>
<td>$-0.001$</td>
<td>0.0277</td>
<td>10.02</td>
<td>0.0019</td>
<td>10.02</td>
<td>0.0019</td>
</tr>
<tr>
<td>$-0.046$</td>
<td>$-0.046$</td>
<td>0.0019</td>
<td>0.0019</td>
<td>0.019</td>
<td>0.0019</td>
<td>0.019</td>
</tr>
<tr>
<td>$0.3069$</td>
<td>$0.3069$</td>
<td>0.0019</td>
<td>0.019</td>
<td>0.0277</td>
<td>0.019</td>
<td>0.0277</td>
</tr>
<tr>
<td>$9.981$</td>
<td>$9.981$</td>
<td>0.0019</td>
<td>0.019</td>
<td>0.0277</td>
<td>0.019</td>
<td>0.0277</td>
</tr>
<tr>
<td>$-0.22$</td>
<td>$-0.22$</td>
<td>0.0019</td>
<td>0.019</td>
<td>0.0277</td>
<td>0.019</td>
<td>0.0277</td>
</tr>
<tr>
<td>$0.1286$</td>
<td>$0.1286$</td>
<td>0.0019</td>
<td>0.019</td>
<td>0.0277</td>
<td>0.019</td>
<td>0.0277</td>
</tr>
<tr>
<td>$10$</td>
<td>$10$</td>
<td>0.0019</td>
<td>0.019</td>
<td>0.0277</td>
<td>0.019</td>
<td>0.0277</td>
</tr>
<tr>
<td>$4.021$</td>
<td>$4.021$</td>
<td>0.0019</td>
<td>0.019</td>
<td>0.0277</td>
<td>0.019</td>
<td>0.0277</td>
</tr>
<tr>
<td>$4 \times 10^{-4}$</td>
<td>$4 \times 10^{-4}$</td>
<td>0.0019</td>
<td>0.019</td>
<td>0.0277</td>
<td>0.019</td>
<td>0.0277</td>
</tr>
</tbody>
</table>

Table 2: Fuzzy rules.
Figure 8: Reference speed and motor speed using an adaptive neural-fuzzy controller.

Figure 9: The result of three types of controllers ANFIS, Fuzzy, and PID.

Figure 10: Reference speed and motor speed using ANFIS, Fuzzy, and PID by halving the (R) and (L) (motor parameters).
by considering the reference speed as a “step.” The simulation results showed that the performance of the control system presented in this paper is very suitable with the reference results [15], especially in the case of load application and parameter changes. Also, in reference [15], 36 fuzzy rules were used, while in our method, a total of 25 rules were defined, and in each training, the number of rules used was less than 10 fuzzy rules.
Figure 14: Reference speed and PMSM speed using ANFIS, Fuzzy, and PID.

Figure 15: A magnification of the transient part of Figure 14.

Figure 16: q-axis current of all three ANFIS, Fuzzy, and PID controllers, in "step" reference input mode.
Figure 17: d-axis current of all three ANFIS, Fuzzy, and PID controllers, in “step” reference input mode.

Figure 18: PWM signal of all three ANFIS controllers, Fuzzy, and PID, in “step” reference input mode.

Figure 19: Zoom part of Figure 18.
Figure 20: PMSM speed control with our proposed method and reference method [15].

Figure 21: Comparison of our proposed method and reference [15] in the case of load apply at $t = 125\, s$ and its removal at $t = 175\, s$.

Figure 22: Magnification of Figure 21 at $t = 125\, s$. 
6. Conclusion

In this paper, a new method for using adaptive fuzzy neural network with combined training method was introduced as a speed controller for permanent magnetic synchronous motor. In this method, the adaptive fuzzy neural network, by identifying and approximating a set of several local controllers, has the ability to control the permanent magnet synchronous motor in a wide operating range and despite the uncertainty and load torque. Each of these local controllers performs well for a particular operating point. The experimental results show that despite the high ability and acceptability of PID control in indeterminate systems, the proposed method performs better because the proposed fuzzy neural network covers not only fuzzy control but also a number of other local controllers. Then, the proposed method was compared with two controllers, Fuzzy and PID, and the results show better performance of the proposed method than the two controllers. In future work, a structure based on type-2 fuzzy systems can be proposed that covers a wider range of uncertainty.

Data Availability

The derived data, models, or codes supporting the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References


