Induction Motor Speed Control with Fast Response using the Levenberg Marquardt method for Electric Cars

Dedid CH

Electrical Department, Institute
Technology of Sepuluh
Nopember
Politeknik Elektronika Negeri
Surabaya
Kampus ITS Sukolilo,
Surabaya, Indonesia

Soebagio

Electrical Department, Institute
Technology of Sepuluh
Nopember
Kampus ITS Sukolilo,
Surabaya, Indonesia

Mauridhi Hery Purnomo

Electrical Department, Institute
Technology of Sepuluh
Nopember
Kampus ITS Sukolilo,
Surabaya, Indonesia

ABSTRACT

Drive system at a low speed electric cars have a hard speed setting is set on a wide range, causing an inconvenience for motorists and a fast response is required any change of speed. It is necessary for good system performance in control motor speed and torque at low speeds which is operated by Field Oriented Control (FOC). In this method of Proportional Integral Derivative used settings Fuzzy Logic Controller (PID-FLC) to dynamically respond to changes in speed and torque in an electric car, so we get smoothness at any speed change and braking as well as maximum torque motor. Neural Network (NN) with Levenberg Marquardt training (LevMar), to set the parameters and the optimal PID gain to drive a three phase induction motor. Test results showed that a fast response to changes in speed electric car.

Keywords

FOC, PID-FLC, Neural Network, LevMar, induction motor electric car.

1. INTRODUCTION

The induction motor has more benefits because it is robust and relatively cheap. It is also mostly used to electrically drive a car in constant speed, big inertia, and no need regular maintenance [1]. However, it also has weakness in complicated speed control. The advance technology in electrics makes this complication easy to solve.

Electric car should be able to move smoothly such as ICE car started from the start having constant acceleration, the running with steady state condition and the breaking of which the car is driven slowly and stopped. To move the car in such way, the motor acceleration needs to be controlled.

Meanwhile an ac motor has a multi-variable nonlinear coupled structure of which its acceleration is difficult to control. However, dc motor has a structure that is decoupled so that its acceleration is easier to control. The control operation of ac motor driver generally needs complicated algorithm implemented in an accurately real time signal process. By using the advance technology of power electronics and electric control, the task for complicated control can be implemented. To do this job, the induction motor is made linear by operating the method of Field Oriented Control [2][3].

In the development of Field Oriented Control, the method is known to have characteristics that is similar to the dc motor induction of which the current magnetic field and anchor current are mutually upright [3].

In the use of drive, electric car is often operated in high or low changing acceleration. And the operation of induction motor on low acceleration often gets in trouble. Therefore, the controller for managing the appropriate operation is needed to get high performance and fast response, control and characteristic paramount, and recovery acceleration as the result of dropped speed due to the load effect. The role of the controller is to improve the response for managing proper acceleration. A common controller is PID that needs recovery act when the load changes. Therefore, the Fuzzy controller based on Self Tuning is implemented.

We ask that authors follow some simple guidelines. In essence, we ask you to make your paper look exactly like this document. The easiest way to do this is simply to download the template, and replace the content with your own material.

2. FIELD ORIENTED CONTROL (FOC)

A motor works on the basis of induction process in the rotor part. When the current flows in the rotor coil, it creates induction that is caused by the differences between the rotor rotation and the field of stator, created by the static coils.

Electromagnetic torque (T_e) is the function of stator current and rotor current, such as:

$$T_e = pM (i_{dr} i_{qs} - i_{qr} i_{ds}) \quad (1)$$

Notes :

M = coupled induction (H)
 i_{ds} = Stator current on axes d (A)
 i_{dr} = Rotor current on axes d (A)
 i_{qs} = Stator current on axes q (A)
 i_{qr} = Rotor current on axes q (A)

Rotor rotation speed is the function from electromagnetic torque, and load torque such as:

$$\frac{J}{p} \frac{d}{dt} \omega_r + K_g \omega_r = T_e - T_l \quad (2)$$

Notes : K_g = Constanta for fraction (kg.m²/dt)
 J = Moment of inersia (kg.m²)
 ω_r = Angular rotor speed (rad/dt)

FOC method consists of controlling stator current in Vector time phase transformation and acceleration in two coordinates d (torque component) and q (flux component).

The basic diagram for FOC in three phases induction motor is :

Figure 1. Control diagram for induction motor

3. NEURAL NETWORK BASED ON LEVENBERG MARQUARDT ALGORITHM

Neural network or artificial neural networks (ANN) is a distributed information processing structure in the form of directed graph. The advantages of this neural network is a network can learn where there are two stages in the operation of the ANN. At this stage of learning to adjust to any provision of input connections to the network produces the desired output with the structure and parameters of the optimal ANN. There are two stages of learning, namely supervised learning (with supervision) and unsupervised learning (without supervision). While the initial testing phase input in the form of the unknown information is given as input to the network. Each cell will perform computation by function aktifasinya connection with the influence of weight gained during the learning process.

3.1 Levenberg Marquardt

Levenberg Marquardt algorithm can be performed using the second derivative approach without having to calculate the Hessian matrix. If the feed forward neural network using the work function of the sum of square hessian matrix can be approximated by [4][5][6] :

$$H = J^T \quad (3)$$

And the gradient can be calculated by :

$$G = J^T e \quad (4)$$

With J is the Jacobian matrix containing first derivatives of the weighting network error and bias network. Levenberg Marquardt algorithm can be calculated with the approach to compute the Hessian matrix with :

$$\Delta X = [J^T J + \mu I]^{-1} J^T e \quad (5)$$

Weighted so that repairs can be calculated :

$$X_{new} = X_{old} + \Delta X \quad (6)$$

Where I is the identity matrix and e is a vector of size pn_o that can be determined by the equation $J^T J$. With the input

dimensions are n_i , hidden dimension is n_h and the output dimension is n_o . So the total weight can be calculated by :

$$W = n_i n_h + n_h n_o + n_h + n_o \quad (7)$$

So the dimension of the Jacobian matrix is pn_o while xw is the Hessian matrix dimensions $w \times w$.

$$J = \begin{bmatrix} \frac{\partial e_{11}(x)}{\partial x_1} & \dots & \frac{\partial e_{11}(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial e_{op}(x)}{\partial x_1} & \dots & \frac{\partial e_{op}(x)}{\partial x_n} \end{bmatrix} \quad (8)$$

 $x =$ weights and biases in the network

Jacobi matrix is a matrix of first derivatives of the weights and the bias error in the network. Jacobi matrix between input layer and hidden layer is a matrix that contains the error derivative of the weights between the input layer, hidden layer and bias.

- Jacobi matrix element between the input layer and hidden layer :

$$\frac{\partial e_o}{\partial v_{i,h}} = \frac{\partial(y_o - t_o)}{\partial y_o} \frac{\partial y_o}{\partial y_{in_o}} \frac{\partial y_{in_o}}{\partial z_h} \frac{\partial z_h}{\partial z_{in_h}} \frac{\partial z_{in_h}}{\partial v_{i,h}} \quad (9)$$

$$\begin{aligned}
&= 1 \cdot f'(y_{in_o}) \frac{\partial(w_{o_o} + \sum_{h=1}^p z_{h w_{h o}})}{\partial z_{h_o}} \cdot f'(z_{in_h}) \cdot \frac{\partial(v_{o h} + \sum_{i=1}^n x_i v_{i h})}{\partial v_{i h}} \\
&= 1 \cdot f'(y_{in_o}) \cdot w_{h_o} \cdot f'(z_{in_h}) \cdot x_i
\end{aligned}$$

- Jacobi matrix element for the bias in the hidden layer

$$\frac{\partial e_o}{\partial v_{(0,n)}} = \frac{\partial(y_o - t_o)}{\partial y_o} \frac{\partial y_o}{\partial y_{in_o}} \frac{\partial y_{in_o}}{\partial z_n} \frac{\partial z_n}{\partial z_{in_n}} \frac{\partial z_{in_n}}{\partial v_{0n}} \quad (10)$$

$$\begin{aligned}
&= 1 \cdot f'(y_{in_o}) \frac{\partial(w_{o0} + \sum_{h=1}^p z_h w_{ho})}{\partial z_h} \cdot f'(z_{in_h}) \cdot \frac{\partial(v_{oh} + \sum_{i=1}^n x_i v_{ih})}{\partial v_{oh}} \\
&= 1 \cdot f'(y_{in_o}) \cdot w_{ho} \cdot f'(z_{in_h}) \cdot 1
\end{aligned}$$

- Jacobi matrix element between the hidden layer and output layer

$$\frac{\partial e_o}{\partial w_{(h,o)}} = \frac{\partial(y_o - t_o)}{\partial y_o} \frac{\partial y_o}{\partial y_{in_o}} \frac{\partial y_{in_o}}{\partial z_h} \quad (11)$$

$$\begin{aligned}
 &= 1 \cdot f'(y_{in_o}) \frac{\partial(w_{0o} + \sum_{n=1}^p z_n w_{no})}{\partial z_n} \\
 &= 1 \cdot f'(y_{in_o}) \cdot w_{no}
 \end{aligned}$$

- Jacobi matrix element for the bias in the output layer

$$\frac{\partial e_o}{\partial w_{0o}} = \frac{\partial(y_o - t_o)}{\partial y_o} \frac{\partial y_o}{\partial y_{in_o}} \frac{\partial y_{in_o}}{\partial w_{0o}} \quad (12)$$

$$= 1 \cdot f'(y_{in_o}) \frac{\partial(w_{0o} + \sum_{h=1}^p z_h w_{ho})}{\partial w_{0o}}$$

$$= 1 \cdot f'(y_{in_o}) \cdot 1$$

4. DESIGN OF SYSTEM SIMULATION

At this stage the hardware design and make the Levenberg Marquardt algorithm for neural network PID tuning that aims to control the plant to match the expected. Here is a block diagram of control system Neural Network with Levenberg Marquardt training method. The diagram in Figure 2 has a data input to the NN and will be split into two input nodes, three hidden nodes and one output node. So the dimension of the matrix jacobinnya to 1 x 13 where "1" is derived from the data input into the NN and "13" comes from the number of weights and biases that are connected from the input layer to hidden layer. So the dimension of the hessian matrix is 13 x 13 which is a product of the J^T with J (see figure 2).

The following are the stages of the control system are assigned the value of NN LevMar for weights and biases are connected. The steps as follow :

- (1) The first step taken to set the initial values of the weights connected between the input and hidden layer (v) and the hidden and output layer (w) are random. Determine the value of μ and β where there are no provisions on how much the value of μ and β , but many studies using $\mu = 0.1$ and $\beta = 10$.
- (2) The second step is performed by calculating the function forward (feed forward). Each hidden node weights summing the input signal as follows :

V_{ih} = the weight between input node i and hidden node h

V_{0h} = value bias in the hidden node h

Calculate the value based on the hidden node activation function sigmoid by :

$$Z_h = f(Z_{in_h}) = \frac{1}{1 + \exp^{-Z_{in_h}}} \quad (14)$$

With :

Z_h = the value of the hidden node h

Each hidden node output signal summing the weights as follows:

$$Y_{out_o} = (Z_h \times W_{ho}) + W_{0o} \quad (15)$$

With :

Y_{out_o} = input signal to the output node o

Z_h = hidden value for the node h

W_{ho} = the value of the weights between the hidden node and output node h

W_{0o} = value of bias on the output node o

Calculating the value of the output node based on the sigmoid activation function :

$$Y_o = f(Y_{out_o}) = \frac{1}{1 + \exp^{-Y_{out_o}}} \quad (16)$$

with:

Y_o = output value node o

Figure 2. Block digaram NN LevMar without PID

$$Z_{in_h} = (X_i \times V_{ih}) + V_{0h} \quad (13)$$

With :

Z_h = the value of the hidden node h

Z_{in_h} = input signal of the hidden node h

X_i = the value of the input node i

- (3) The third step is done by calculating the value of error where the error function approximated by Sum Square Error (SSE). Where to compare the value of each output (Y_o) with a target (t_k) with the following equation:

$$SSE = \frac{1}{2} \sum_{o=1}^p (t_o - Y_o)^2 \quad (17)$$

(4) The fourth step is done by calculating the Jacobian matrix (J) which contains the first error derivative of the weights and bias.

(5) The fifth step is done by calculating the difference in weight by using equation (18) is :

$$\Delta X = [J^T J + \mu I]^{-1} J^T e \quad (18)$$

With :

$J^T J$ = an approximation of the Hessian matrix

$J^T e$ = the gradient of the error

(6) The sixth step is done by calculating a new weight to the equation:

$$X_{new} = X_{old} + \Delta X \quad (19)$$

Having obtained the new weights are calculated as the error back to step 3. If the error is reduced, the new do (μ/β) and return to step 2 to step 7. If a new error is not reduced then do ($\mu \times \beta$) and return to step 5. This will be carried out continuously until the same error with the error limit. LevMAR Neural Network algorithm as a whole are as follows on figure 3.

5. SIMULATION RESULTS AND ANALYSIS

In the simulation, induction motor run with Neural Network with Levenberg Marquardt training. The simulation results are compared between using PID and without PID Controller. In the simulation results with the Levenberg Marquardt without PID controller simulation results obtained are fast response time, but still indicate the presence of overshoot (shown in the figure 4). If the simulation with Levenberg Marquardt training with PID Controller simulation results obtained are fast response time and almost did not show any overshoot (shown in the figure 6). To demonstrate the occurrence of motor braking many times, done by changing a few different speeds vary by varying the set point value, then the simulation results still indicate the presence of overshoot at maximum speed, but at lower speeds the simulation results indicate that the smaller overshoot (shown in the figure 5).

Figure 4. Simulation result using Levenberg Marquardt without PID controller with set point 1700 rpm

Figure 4. Flowchart of Neural Network LevMar without PID Fuzzy controller with set point changes

From the above data it was found that a considerable overshoot occurs when there is a significant change in set point as shown in "B", but when the low set point change in the overshootnya very small as shown in the "A"

Figure 5 Simulation result using Levenberg Marquardt with PID Fuzzy controller with set point 500 rpm

Figure 6 Simulation result using Levenberg Marquardt with PID Fuzzy controller with set point 1400 rpm

6. CONCLUSION

From the results of simulation using Visual Basic, a summary can be presented as follows :

- The simulation of motor using PID Fuzzy controller and NN LevMar does not show overshoot results obtained so the conditions are like on a first-order system.
- In high frequency operation, the control method can produce better percent overshoot.
- Steady state error is relatively very small on the examination of some set points.

7. REFERENCE

- [1] C. C. Channand K. T. Chau, *Member, IEEE*, An Overview of Power Electronics in Electric Vehicles, *IEEE Transactions on Industrial Electronics*, vol. 44, no. 1, February 1997.
- [2] Nobuyoshi Mutoh, Satoru Kaneko, Taizou Miyazaki, Ryosou Masaki, and Sanshiro Obara, A Torque Controller Suitable for Electric Vehicles, *IEEE Transactions On Industrial Electronics*, vol. 44, no. 1, February 1997
- [3] Mariam Khan, and Narayan C. Kar, Performance Analysis of Fuzzy Based Indirect Field Oriented Control of Induction Motor Drives for Hybrid Electric Vehicles, *IEEE*,
- [4] Martin T. Hagan and Mohamad B. Menhaj, Training Feedforward with the Marquardt Algorithm, *IEEE*

Transaction on Neural Network, Vol. 5 No. 6 November 1994.

- [5] Manolis I. A. Lourakis, A Brief Description of the Levenberg-Marquardt Algorithm Implemented by levmar, Institute of Computer Science Foundation for Research and Technology - Hellas (FORTH) Vassilika Vouton, P.O. Box 1385, GR 711 10 Heraklion, Crete, GREECE, February 11, 2005
- [6] Sam Roweis. Levenberg-Marquardt Optimization[J].
- [7] Jae-Sung Yu, Sang-Hoon Kim, Fuzzy logic based Vector Control scheme for permanent magnet Synchronous motors in elevator drive application, *IEEE Trans on Industrial Electronics*, Vol. 54 No. 4, Aug 2007.
- [8] Dynamic Simulation of electric Machinery, Using Matlab/Simulink, Chee-Mung On, Prentice Hall PTR, Upper Saddle River, New Jersey 07458, 1998.
- [9] Mohamed Rachid Chekkouri, Jordi Catal, Fuzzy Adaptive Control of an Induction Motor Drive, *ISSN 0005-1144 ATKAAF* 44(3-4), 113-122 (2003)
- [8] M. Vasudevan, R. Arumugam, S. Paramasivam, High Performance Adaptive Intelligent Direct Torque Control Schemes for Induction Motor Drives, *Serbian Journal Of Electrical Engineering*, Vol. 2, No. 1, May 2005, 93 – 116.
- [9] Mamdani, E.H, Application Of Fuzzy Algorithms For Control Of Simple Dynamic Plant, *Proc. Of IEEE*, 12, 1974, h. 1585 – 1588.
- [10] CC Lee, Fuzzy Logic In Control System : Fuzzy Logic Controller Part I, *IEEE Trans. On Systems, Man, And Cybernetics*, Vol 20 No 2, March – April 1990.
- [11] JF Hurdle, The Synthesis Of Compact Fuzzy Neural Circuits, *IEEE Trans. Fuzzy Systems*, Vol. 5, No. 1, February 1997, p. 44
- [12] W. Li, A Method for Design of a Hybrid Neuro-Fuzzy Control Systems Based On Behavior Modeling, *IEEE Trans. Fuzzy Systems*, Vol 5, No 1, February 1997, p. 128.
- [13] Z. Q. Liu and F. Yan, Fuzzy Neural Network in Case Based Diagnostic Systems, *IEEE Trans. Fuzzy Systems*, Vol 5, No 2, May 1997, p. 209

AUTHORS PROFILE

¹ Department of Electrical Engineering, Faculty of Industrial Technology, Institute of Technology Sepuluh Nopember Sukolilo Campus, Arief Rahman Hakim Road, Surabaya-Indonesia, 60111 and Engineering Electronic Polytechnic in Surabaya (PENS) Sukolilo Campus, Arief Rahman Hakim Road, Surabaya-Indonesia, 60111.

² Department of Electrical Engineering, Faculty of Industrial Technology, Institute of Technology Sepuluh Nopember Sukolilo Campus, Arief Rahman Hakim Road, Surabaya-Indonesia, 60111.

³ Department of Electrical Engineering, Faculty of Industrial Technology, Institute of Technology Sepuluh Nopember Sukolilo Campus, Arief Rahman Hakim Road, Surabaya-Indonesia, 60111.

Dedid CH, born in Pasuruan, Indonesia, December 27, 1962. Educational back-grounds:

Engineer in Electrical Engineering Institute of Technology Sepuluh Nopember Surabaya, Surabaya Indonesia (1986).

MT Electrical Engineering Institute of Technology Sepuluh Nopember Surabaya, Surabaya Indonesia (2002)

Post-graduate student in Electrical Engineering, in Institute of Technology Sepuluh Nopember Surabaya-Indonesia (2007-ow) Some papers were published in Journals and seminars by colaboration with Prof. Mauridhi Hery Purnomo and Prof Soebagio.

Soebagio, born in Malang, Indonesia, February 28, 1941. He graduated from Electrical Engineering Institute of Technology Bandung, Bandung Indonesia. Now, he is a professor in electric drive system in Department of Electrical Engineering, Institute of Technology Sepuluh Nopember Surabaya

Indonesia. Prof. Soebagio is a member of Indonesian Institution of Engineers.

Mauridhi Hery Purnomo, born in Bangkalan, Indonesia, September 16, 1958. He graduated from Electrical Engineering, in Institute of Technology Sepuluh Nopember Surabaya-Indonesia. Now, he is a professor in intelligence system in Department of Electrical Engineering, Institute of Technology Sepuluh Nopember Surabaya Indonesia. Prof. M.H. Purnomo is a member of Indonesian Institution of Engineers.