Vietnam Journal of Science and Technology ***59*** *(2) (2021) 234-248*

doi:10.15625/2525-2518/59/2/14921

USING RADIAL BASIS FUNCTION NEURAL NETWORK FOR PMSM TO OVERCOME THE ABRUPT LOAD CHANGE AND UNCERTAINTIES

Nguyen Vu Quynh

Department of Electrical and Electronics, Lac Hong University, 810000, Dong Nai, Viet Nam

Email: *vuquynh@lhu.edu.vn*

Received: 24 March 2020; Accepted for publication: 11 February 2021

Abstract. This paper presents a new method using Radial Basis Function Neural Network (RBFNN) for Permanent magnet synchronous motor (PMSM) to overcome the abrupt load change and uncertainties. Firstly, a mathematical model of the PMSM drive is derived; a Fuzzy PI controller in which an RBFNN is used to adjust its parameters is applied to the speed controller for coping with the system's effect dynamic uncertainty and the external load. Secondly, the Very high-speed integrated circuit Hardware Description Language (VHDL) is adopted to describe the speed control IC behavior, including the circuits of space vector pulse width modulation (SVPWM), coordinate transformation, RBFNN, and Fuzzy PI. Thirdly, the simulation work is implemented by MATLAB/Simulink and ModelSim co-simulation mode. The PMSM, inverter, and speed command are performed in Simulink, and the speed controller of the PMSM drive is executed in ModelSim. Finally, the co-simulation results validate the effectiveness of the proposed algorithm-based PMSM speed control system.

Keywords: VHDL, Simulink, Simulation, Fuzzy PI, Neural Network.

Classification numbers: 4.7.4, 4.10.3, 4.10.4.

1. INTRODUCTION

The Permanent magnet synchronous motor (PMSM) has been increasingly used in many automation control fields as actuators, due to its advantages of superior power density, high-performance motion control with fast speed, and better accuracy. However, in industrial applications, there are many uncertainties, such as system parameter uncertainty, external load disturbance, friction force, unmodeled uncertainty which always diminishes the performance quality of the pre-design of the motor driving system. To cope with this problem, in recent years, many intelligent control techniques, such as proportional integral derivative (PID) control [1 - 2], fuzzy control [3 - 4], adaptive fuzzy control [5 - 6], neural networks control [7 - 8], and other control methods [9 - 10], have been developed and applied to the speed of the motor to obtain high operating performance. The PMSM is sensitive to the dynamic uncertainty system and the external load. The PI or Fuzzy controller is widely used to control motor speed; however, it is challenging to get excellent performance. When a load of rotor changes, it is hard to guarantee that the system runs stably. This research did with master-slave motor control to ensure the motor's stability in changing load [11]. Although fuzzy control has been successfully applied in several industrial automation applications, however, it is not an easy task to obtain an optimal set of fuzzy membership functions and rules in fuzzy controller (FC). In this work, a Radial Basis Function Neural Network (RBFNN) is used to identify the plant dynamic and provided more accurate plant information for parameter tuning of the Fuzzy PI controller. Various successful RBFNN applications in different fields such as signal modeling, non-linear time series forecast, recognition of dynamic systems, pattern identification, and knowledge learning are shown [12]. In the implementation of the proposed algorithm, although the algorithm's execution requires many computations, Field Programmable Gate Array (FPGA) can provide a solution to this issue. Especially, the FPGA with the programmable hard-wired feature, fast computation ability, shorter design cycle, embedding processor, low power consumption, and higher density is better for the implementation of the digital system [13 - 16] than digital signal processor (DSP).

The co-simulation work by Electronic Design Automation (EDA) Simulator Link has been gradually applied to verify the effectiveness of the Verilog and VHDL code in the motor drive system [17 - 18]. The EDA Simulator Link [19] provides a co-simulation interface between MATLAB, Simulink, and hardware description language (HDL) simulators-ModelSim [20]. The researcher can verify a VHDL, Verilog, or mixed-language implementation against the Simulink model or MATLAB algorithm [19]. Therefore, the EDA Simulator Link lets the researcher use MATLAB code and Simulink models as a test bench that generates stimulus for an HDL simulation and analyzes the simulation's response [19]. In this paper, the co-simulation by the EDA Simulator Link is applied. The PMSM, inverter, and speed command are performed in Simulink, and the speed control IC described by VHDL code is executed in ModelSim. Finally, some simulation results validate the effectiveness of the proposed Neural Network Fuzzy PI Controller (NFPC) based speed control system of the PMSM drive. The novelty of this research is in using RBFNN to adjust the parameters of the Fuzzy PI controller. This method applied to the speed controller for coping with the effect of the system dynamic uncertainty and the external load. The co-simulation results have been compared between two kinds of the controller and, it validated the effectiveness of the proposed algorithm-based PMSM speed control system.

2. THE DESIGN OF NFPC FOR PMSM DRIVE

2.1. Vector control for PMSM drive

2.1.1 The PMSM mathematical model:

The mathematical model of PMSM on d and q axis

$$\frac{di_d}{dt}=\frac{1}{L_d}v_d-\frac{R}{L_d}i_d+\frac{L_q}{L_d}p*\omega_r*i_q \tag{1}$$

$$\frac{di_q}{dt}=\frac{1}{L_q}v_q-\frac{R}{L_q}i_q-\frac{L_d}{L_q}p*\omega_r*i_d-\frac{\lambda*p*\omega_r}{L_q} \tag{2}$$

in which: L_q and L_d are the inductance on q and d axis; R is the resistance of the stator windings; i_q and i_d are the currents on q and d axis; V_q and V_d are correspondingly the voltage on q and d axis; λ is the permanent magnet flux linkage; p is a number of pole pairs; ω_r is the rotational speed of the rotor.

2.1.2 The vector controller

The current controller of the PMSM drive in Figure 1 is based on a vector control approach. The current controller structure includes Clark, Park, invert Park, Invert Clark transformation, two PID controllers, and sine/cosine of flux angle. Three-phases current is fed back through vector control, enabling controlling current $i_d \approx 0$. It helped control a three-phase motor similar to a DC motor. The torque of the PMSM is controlled via current on the q axis (i_q). The vector control describes below:

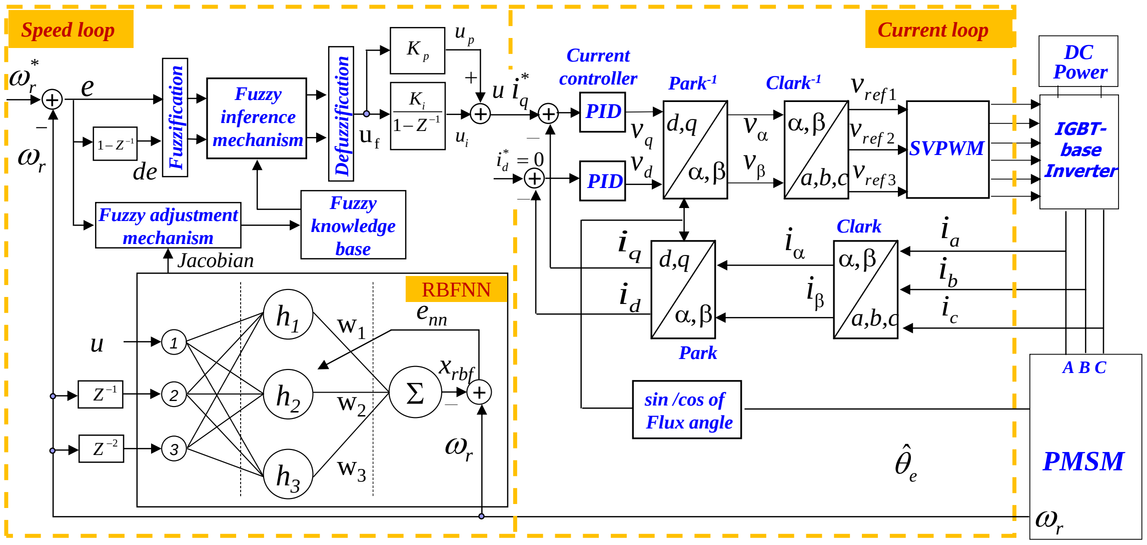


Figure 1. The block diagram of the PMSM controller.

The Clark transformation: stationary i_a, i_b, i_c frame to stationary i_α, i_β frame.

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & \frac{-1}{3} & \frac{-1}{3} \\ 0 & \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{3}$$

The Park transformation: the stationary i_α, i_β frame transform to synchronously rotating i_d, i_q frame.

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \cos\theta_e & \sin\theta_e \\ -\sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \tag{4}$$

The invert Park transformation: synchronously rotating v_d, v_q frame transform to stationary v_α, v_β frame.

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \begin{bmatrix} \cos\theta_e & -\sin\theta_e \\ \sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} v_d \\ v_q \end{bmatrix} \tag{5}$$

The invert Clark transformation: stationary v_α, v_β frame transform to stationary v_{ref1}, v_{ref2}, v_{ref3} (a-b-c) frame.

$$\begin{bmatrix} v_{ref1} \\ v_{ref2} \\ v_{ref3} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{-1}{2} & \frac{\sqrt{3}}{2} \\ \frac{-1}{2} & \frac{-\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} v_{\alpha} \\ v_{\beta} \end{bmatrix} \tag{6}$$

The following formula calculates the electromagnetic moment of the motor:

$$T_e = 1.5p[\lambda * i_q + (L_d - L_q) i_d * i_q] \approx K_t * i_q \tag{7}$$

Mathematical equations of the motor when the load-bearing:

$$\frac{d\omega_r}{dt} = \frac{1}{J_m(T_e - F\omega_r - T_m)} \tag{8}$$

where: T$_e$ is the motor torque; J$_m$ and F are the inertia and viscous friction of rotor and load, respectively; T$_m$ is the shaft mechanical torque, K$_t$ is the torque constant.

2.2. Design of Fuzzy PI Controller

The Fuzzy controller in this study uses a singleton fuzzifier, triangular membership function, product-inference rule, and central average defuzzifier method. In Figure 1, the tracking error e and the error change de are defined by

$$e(k) = \omega_r^*(k) - \omega_r(k) \tag{9}$$

$$de(k) = e(k) - e(k-1) \tag{10}$$

where u_f is the output of the fuzzy controller, and ω_r^* is the setting value. The design of the Fuzzy PI controller is as follows. Firstly, in Figure 1, the *e* and *de* are taken as the input variable of FC, and their linguist variables are defined as *E* and *dE*. Each linguist value of *E* and *dE* is based on the symmetrical triangular membership function. Secondly, the computation of the membership degree for *e* and *de* are done. However, the only two linguistic values can be excited in any input value. Thirdly, the selection of the initial fuzzy controller rules refers to the dynamic response characteristics, such as,

$$IF\ e\ is\ A_i\ and\ \Delta e\ is\ B_j\ THEN\ u_f\ is\ c_{j,i}\,, \tag{11}$$

where i and j are from 0 to 6, A_i and B_j are fuzzy numbers, and $c_{j,i}$ is the real number. Finally, to construct the fuzzy system $u_f(e,de)$, the singleton fuzzifier, product-inference rule, and central average defuzzifier method is adopted, and the following expression can obtain the output of the fuzzy inference:

$$u_f(e,de) = \frac{\sum_{n=i}^{i+1} \sum_{m=j}^{j+1} c_{m,n} [\mu_{A_n}(e) * \mu_{B_m}(de)]}{\sum_{n=i}^{i+1} \sum_{m=j}^{j+1} \mu_{A_n}(e) * \mu_{B_m}(de)} \underline{\underline{\Delta}} \sum_{n=i}^{i+1} \sum_{m=j}^{j+1} c_{m,n} * \xi_{n,m} \tag{12}$$

where $\xi_{n,m} \underline{\underline{\Delta}} \mu_{A_n}(e) * \mu_{B_m}(de)$.

The PI controllers are combined with the Fuzzy controller for the speed loop of the PMSM drive (Figure 1). The digital function of it is shown below:

$$\mathrm{u}_p(k) = K_p * \mathrm{u}_f(k) \tag{13}$$

$$\mathrm{u}_i(k) = \mathrm{u}_i(k-1) + K_i * \mathrm{u}_f(k-1) \tag{14}$$

$$\mathrm{u}(k) = \mathrm{u}_p(k) + \mathrm{u}_i(k) \tag{15}$$

where the u_f is the output of the Fuzzy controller, K_p and K_i. are the gain of the PI controller.

2.3. Design of Radial Basis Function Neural Network

The RBFNN adopted here is a three-layer architecture, which is shown in Figure 1 and comprised of one input layer, one hidden layer, and one output layer.

The RBFNN has three inputs by $\mathrm{u}(k), \omega_r(k-1), \omega_r(k-2)$, and its vector form is represented by

$$X = [u(k), \omega_r(k-1), \omega_r(k-2)]^T \tag{16}$$

Furthermore, the multivariate Gaussian function is used as the activated function in the hidden layer of RBFNN, and its formulation is shown as follows.

$$h_r = \exp\left(-\frac{\|X - c_r\|^2}{2\sigma_r^2}\right), r = 1,2,3,4,....q \tag{17}$$

where $c_r = [c_{r1}, c_{r2}, c_{r3}]^T$ and σ_r denote the node center, and node variance of r^{th} neuron, and $\|X - c_r\|$ is the norm value, which is measured by the inputs and the node center at each neuron. And the network output of the RBFNN block in Figure 1 can be written as

$$\omega_{rbf} = \sum_{r=1}^{q} w_r * h_r \tag{18}$$

where ω_{rbf} is the output value; w_r and h_r are the weight and output of r^{th} neuron, respectively.

The instantaneous cost function is defined as follows.

$$J = \frac{1}{2}(\omega_{rbf} - \omega_r)^2 \underline{\underline{\Delta}} \frac{1}{2} e_{nn}^2 \tag{19}$$

According to the gradient descent method, the learning algorithm of weights, node center, and variance are as follows:

$$w_r(k+1) = w_r(k) + \eta * e_{nn}(k) * h_r(k) \tag{20}$$

$$c_{rs}(k+1) = c_{rs}(k) + \eta * e_{nn}(k) * w_r(k) * h_r(k) \frac{X_s(k) - c_{rs}(k)}{\sigma_r^2(k)} \tag{21}$$

$$\sigma_r(k+1) = \sigma_r(k) + \eta * e_{nn}(k) * w_r(k) * h_r(k) \frac{\|X(k) - c_r(k)\|^2}{\sigma_r^3(k)} \tag{22}$$

where r = 1,2,..q, s = 1,2,3 and η is a learning rate. Further, the $\frac{\partial \omega_r}{\partial i_q^*}$ is Jacobian transformation and can be derived from the RBFNN block in Figure 1:

$$\frac{\partial \omega_r}{\partial u} \approx \frac{\partial \omega_{rbf}}{\partial u} = \sum_{r=1}^{q} w_r * h_r \frac{c_{r1} - u(k)}{\sigma_r^2} \tag{23}$$

2.4. Adjusting mechanism of fuzzy PI controller

The gradient descent method is used to derive the NFPC control law in Figure 1. The adjusting mechanism of Fuzzy PI parameters is to minimize the square error between the rotor speed and the input setting value. The instantaneous cost function is firstly defined by

$$J_e \underline{\Delta} \frac{1}{2} e^2 = \frac{1}{2} (\omega_r^* - \omega_r)^2 \tag{24}$$

and the parameters of $c_{m,n}$ are adjusted according to

$$\Delta c_{m,n} \propto -\frac{\partial J_e}{\partial c_{m,n}} = -\alpha \frac{\partial J_e}{\partial c_{m,n}} \tag{25}$$

where α represents the learning rate. Secondly, the chain rule is used, and the partial differential equation for J_e in (18) can be written as

$$\frac{\partial J_e}{\partial c_{m,n}} = -e \frac{\partial \omega_r}{\partial u_f} \frac{\partial u_f}{\partial c_{m,n}} \tag{26}$$

Further, from (9) and using the Jacobian formulation from (17), we can respectively get

$$\frac{\partial u_f(k)}{\partial c_{m,n}(k)} = d_{n,m} \tag{27}$$

$$\frac{\partial \omega_r}{\partial u_f} \approx (K_P + K_i) \frac{\partial \omega_{rbf}}{\partial u} = (K_P + K_i) \sum_{r=1}^{q} w_r * h_r \frac{c_{r1} - u(k)}{\sigma_r^2} \tag{28}$$

Therefore, substituting (21) and (22) into (20), the parameters $c_{m,n}$ of the Fuzzy PI controller described in (9) can be adjusted by the following expression.

$$\Delta c_{m,n}(k) = \alpha * e(k) * (K_p + K_i) * d_{n,m} \sum_{r=1}^{q} w_r * h_r \frac{c_{r1} - u(k)}{\sigma_r^2} \tag{29}$$

with $m = j, j+1$ and $n = i, i+1$.

2.5. The implement of algorithm on FPGA

The internal architecture of the proposed speed controller for the PMSM drive is shown in Figure 2. The inputs of this controller are input speed command ω_m, rotor speed ω_r, flux angle θ_e, measured three-phase currents (i_a, i_b, i_c), and the output is PWM command. The speed controller for PMSM mainly includes an NFPC-based speed controller (Speed loop), a current controller (Current loop). The sampling frequency of current and speed control is designed with 16 kHz and 2 kHz, respectively. The input clock is 50 MHz (CLK signal), 25 MHz (CLK_40n signal), clock to supply all modules of the speed controller. All modules in Figure 2 are described by VHDL and simulated in ModelSim. The Finite state machine (FSM) is employed to model the NFPC-based speed controller in Figure 2, and it is shown in Figures 3-5, which uses adders, multipliers, and registers, etc. It takes manipulates 85 steps and 24 steps machine to carry out the NFPC and Current loop controller computation, respectively. Although the algorithm of the

NFPC is high complexity, the FSM can give adequate modeling and easily be described by VHDL.

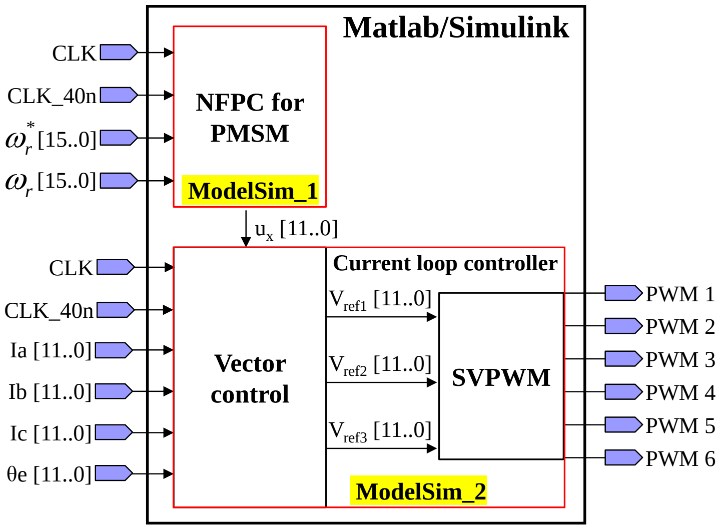


Figure 2. The internal circuit of the proposed speed control IC for PMSM drive.

In Figure 3, 85 steps are carried out the Fuzzy PI controller in which an RBFNN adjusts its parameters. The steps s_0-s_1 compute the rotor speed error and error change. The steps s_2-s_5 compute the fuzzification. The step S6 look-up fuzzy table to select four values of c_{mn} bassed-on i and j got from s_5. The steps s_7-s_{15} compute the defuzzification function; those are the equations 6-11. The steps s_{16}-s_{18} compute the current command, and it is the PI function. The steps s_{19}-s_{70} and s_{71}-s_{73} calculate three neural networks; the FPGA is parallel processing; Therefore, it need only 52 steps to finish. Finally, the steps s_{74}-s_{84} turn the Fuzzy rule parameters.

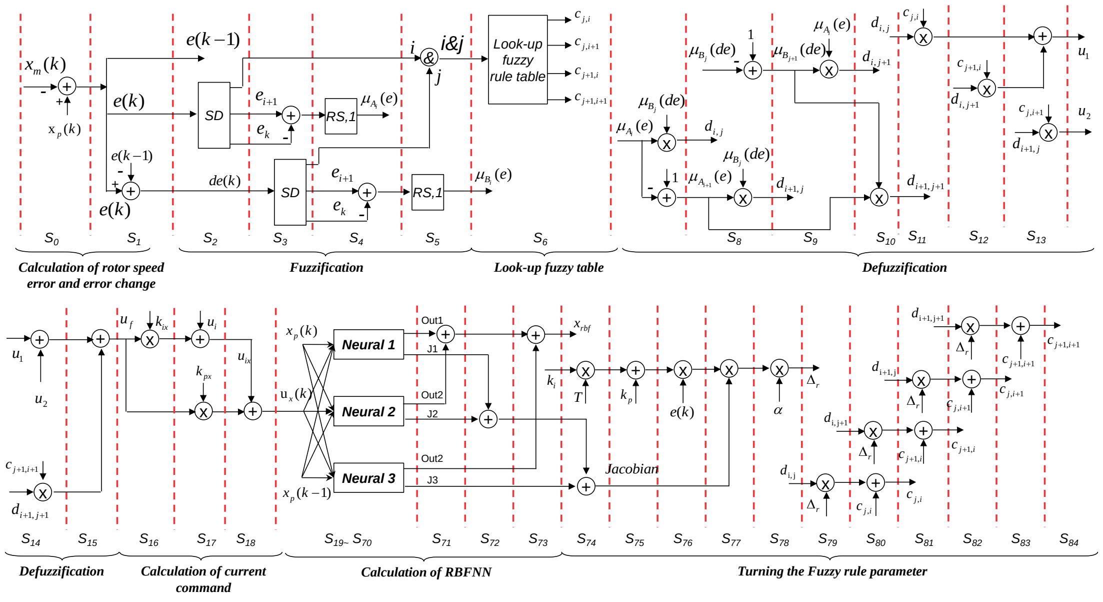


Figure 3. The internal circuit of NFPC speed controller.

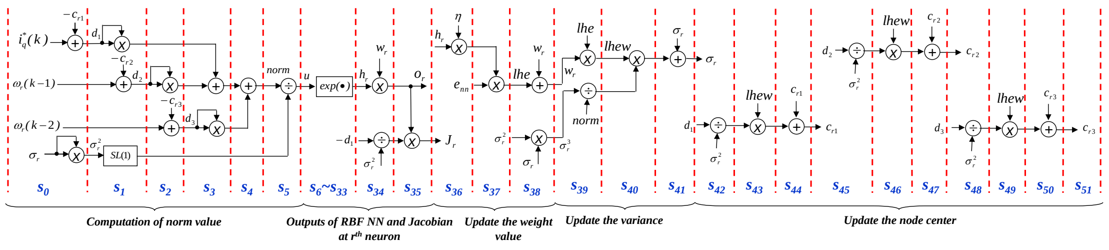

Computation of norm value
Outputs of RBF NN and Jacobian at rth neuron
Update the weight value
Update the variance
Update the node center

Figure 4. The internal circuit of one neural network.

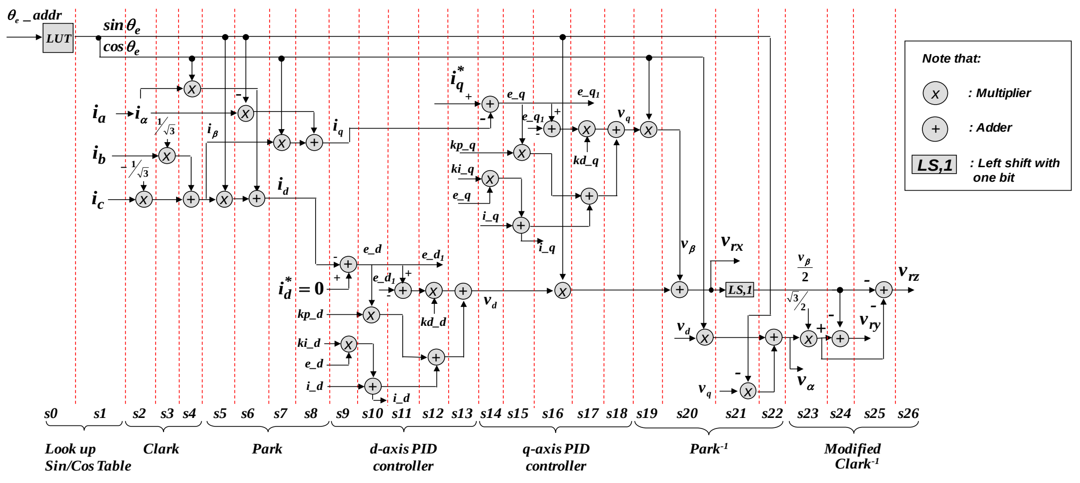


Figure 5. The internal circuit current control and coordinate transform.

In Figure 4, to compute the one neural, it takes 52 steps. The steps s_0-s_5 compute the norm value. The steps s_6-s_{33} calculate the exponential function (equation 17); The step s_{34}-s_{35} compute the output of neural and Jacobian at rth neural. Finally, the steps s_{36}-s_{51} update the weight, variance, and node center value of the neural network.

In Figure 5, there are 27 steps to carry out the current vector controller for PMSM. The steps s_0-s_1 look-up table to take the value of sine and cosine functions. The steps s_2-s_8 calculate the Clark and Park transformations. The PID controller turns the current on d and q-axis from step s_9-s_{18}. Finally, the steps s_{19}-s_{26} calculate the invert Park and invert Clark transformations.

The operation of each step in Figures 3-5 can be completed within 40 ns (25 MHz clock); therefore, a total of 85 steps of NFPC and 27 steps of the Current loop controller only need 3.4 µs and 1.08 µs operational times, respectively. The FPGA resource usages of the Current loop controller and NFPC in Figure 2 are 36,701 LEs and 174,473 RAM bits.

3. THE SIMULATION RESULTS

The co-simulation architecture for NFPC-based speed control of the PMSM drive is shown in Figure 6. The PMSM, inverter, and speed command are implemented in the Simulink program, and the speed controller of the PMSM drive is realized in the VHDL code using the ModelSim program. The SimPowerSystem blockset in the Simulink executes the PMSM and the inverter. The EDA simulator link for ModelSim executes the co-simulation using VHDL code running in the ModelSim program. The designed PMSM parameters used in the simulation of Figure 6 are that pole pairs is 4, stator phase resistance is 1.3 Ω, stator inductance is 6.3 mH, inertia is J = 0.000108 kg*m^2, and friction factor is F = 0.0013 N*m*s.

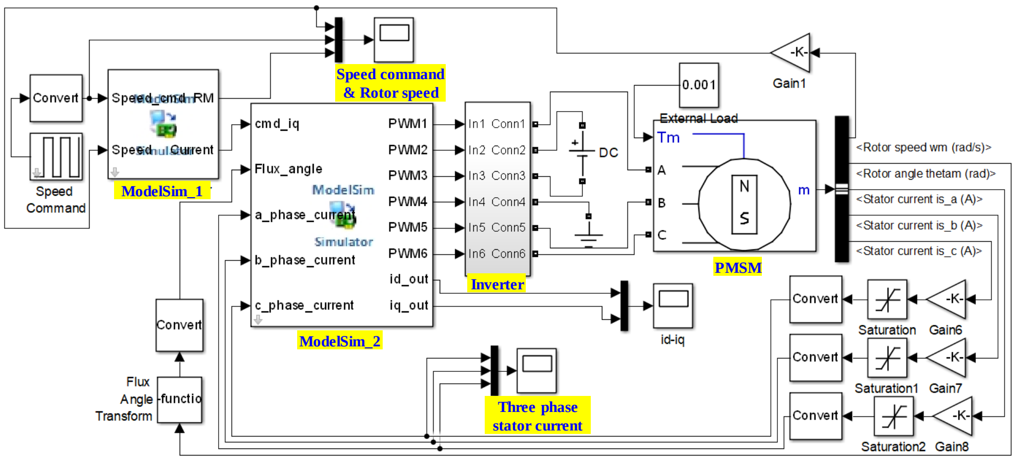


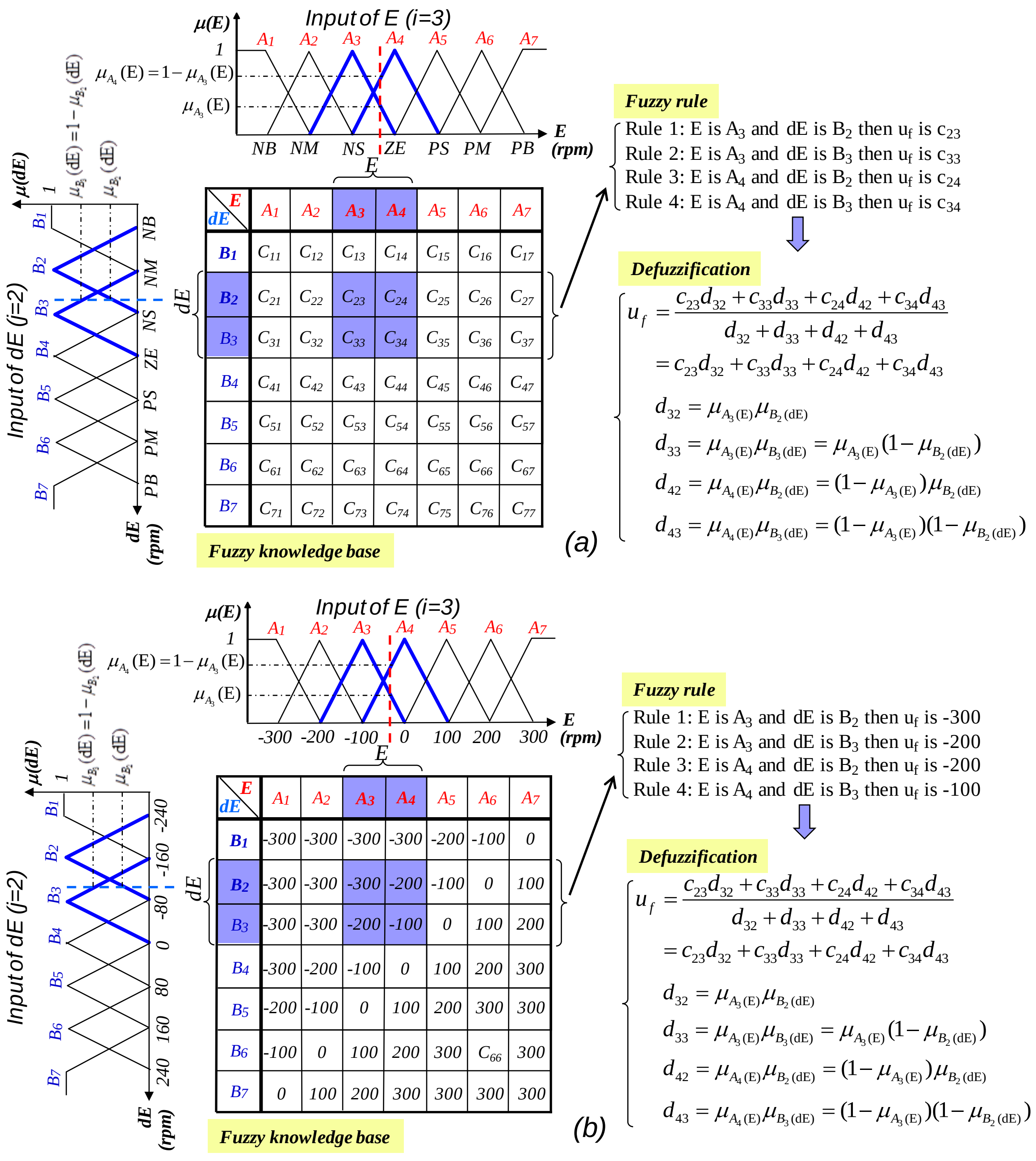
Figure 6. The Simulink model of the PMSM speed controller.

The simulation is used to evaluate the effectiveness of the proposed control algorithm. Three tested cases with different PMSM parameters are considered to evaluate the controller performance, in which Normal-load condition with J = 0.000108, F = 0.0013; light-load condition with 0.000036, F = 0.00043; Heavy-load condition with J = 0.000324, F = 0.0039.

The co-simulation is carried out in Figure 6. The control objective is to control the rotor speed of PMSM to track the input setting value. In the Fuzzy PI controller design, the membership function, and the Fuzzy PI rule table are designed as Figure 7a. In the NFPC design, except that the parameters of the $c_{i,j}$ in Figure 7a can be tuned using (29); others are the same as the Fuzzy PI controller. The initial parameters of the Fuzzy controller are shown in Figure 7b. Herein, the learning rate α is set as 0.03. A squared wave with a period of 0.05 seconds and magnitude of 300 rpm is used as a tested input command. To compare the tracking performance of the aforementioned two controllers at various system conditions, the system parameters are initially designed at the normal-load condition, and the controller is adopted by the Fuzzy PI controller only, and the simulation result is shown in Figure 8, which presents a good following response. However, when the system parameters change to the light-load and heavy-load condition, the results in Figures 9-10 show that the response worsens with oscillation occurred in light-load condition and, the slow response occurs in heavy-load condition. To cope with this problem, an NFPC is adopted, and its simulation results are shown in Figures 11-12. Due to the $c_{i,j}$ parameters in the Fuzzy PI controller can be tuned to reduce the error between the rotor speed and the input setting value; the rotor speed can accurately track well in Figures 11-12. Therefore, the simulation results in Figures 8-12 demonstrate that the proposed NFPC-based speed control IC for PMSM drive is effective and robust. The flowchart of the proposed algorithm is shown in Figure 13.

4. CONCLUSION

This study presented an NFPC-based speed controller for PMSM drive and successfully demonstrated its performance through co-simulation using Simulink and ModelSim. After confirming the effectiveness of the VHDL code of speed controller, we will realize this code in the experimental FPGA-based PMSM drive system for further verifying its function in future work.



(a)

Fuzzy knowledge base

dE \ E	A1	A2	A3	A4	A5	A6	A7
B1	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}
B2	C_{21}	C_{22}	C_{23}	C_{24}	C_{25}	C_{26}	C_{27}
B3	C_{31}	C_{32}	C_{33}	C_{34}	C_{35}	C_{36}	C_{37}
B4	C_{41}	C_{42}	C_{43}	C_{44}	C_{45}	C_{46}	C_{47}
B5	C_{51}	C_{52}	C_{53}	C_{54}	C_{55}	C_{56}	C_{57}
B6	C_{61}	C_{62}	C_{63}	C_{64}	C_{65}	C_{66}	C_{67}
B7	C_{71}	C_{72}	C_{73}	C_{74}	C_{75}	C_{76}	C_{77}

Fuzzy rule

Rule 1: E is A_3 and dE is B_2 then u_f is c_{23}
Rule 2: E is A_3 and dE is B_3 then u_f is c_{33}
Rule 3: E is A_4 and dE is B_2 then u_f is c_{24}
Rule 4: E is A_4 and dE is B_3 then u_f is c_{34}

Defuzzification

$$\begin{cases} u_f = \dfrac{c_{23}d_{32} + c_{33}d_{33} + c_{24}d_{42} + c_{34}d_{43}}{d_{32} + d_{33} + d_{42} + d_{43}} \\ \quad = c_{23}d_{32} + c_{33}d_{33} + c_{24}d_{42} + c_{34}d_{43} \\ d_{32} = \mu_{A_3(E)}\mu_{B_2(dE)} \\ d_{33} = \mu_{A_3(E)}\mu_{B_3(dE)} = \mu_{A_3(E)}(1 - \mu_{B_2(dE)}) \\ d_{42} = \mu_{A_4(E)}\mu_{B_2(dE)} = (1 - \mu_{A_3(E)})\mu_{B_2(dE)} \\ d_{43} = \mu_{A_4(E)}\mu_{B_3(dE)} = (1 - \mu_{A_3(E)})(1 - \mu_{B_2(dE)}) \end{cases}$$

(b)

Fuzzy knowledge base

dE \ E	A1	A2	A3	A4	A5	A6	A7
B1	-300	-300	-300	-300	-200	-100	0
B2	-300	-300	-300	-200	-100	0	100
B3	-300	-300	-200	-100	0	100	200
B4	-300	-200	-100	0	100	200	300
B5	-200	-100	0	100	200	300	300
B6	-100	0	100	200	300	C_{66}	300
B7	0	100	200	300	300	300	300

Fuzzy rule

Rule 1: E is A_3 and dE is B_2 then u_f is -300
Rule 2: E is A_3 and dE is B_3 then u_f is -200
Rule 3: E is A_4 and dE is B_2 then u_f is -200
Rule 4: E is A_4 and dE is B_3 then u_f is -100

Defuzzification

$$\begin{cases} u_f = \dfrac{c_{23}d_{32} + c_{33}d_{33} + c_{24}d_{42} + c_{34}d_{43}}{d_{32} + d_{33} + d_{42} + d_{43}} \\ \quad = c_{23}d_{32} + c_{33}d_{33} + c_{24}d_{42} + c_{34}d_{43} \\ d_{32} = \mu_{A_3(E)}\mu_{B_2(dE)} \\ d_{33} = \mu_{A_3(E)}\mu_{B_3(dE)} = \mu_{A_3(E)}(1 - \mu_{B_2(dE)}) \\ d_{42} = \mu_{A_4(E)}\mu_{B_2(dE)} = (1 - \mu_{A_3(E)})\mu_{B_2(dE)} \\ d_{43} = \mu_{A_4(E)}\mu_{B_3(dE)} = (1 - \mu_{A_3(E)})(1 - \mu_{B_2(dE)}) \end{cases}$$

Figure 7. Fuzzy membership function (a) and the initial Fuzzy PI rule table (b).

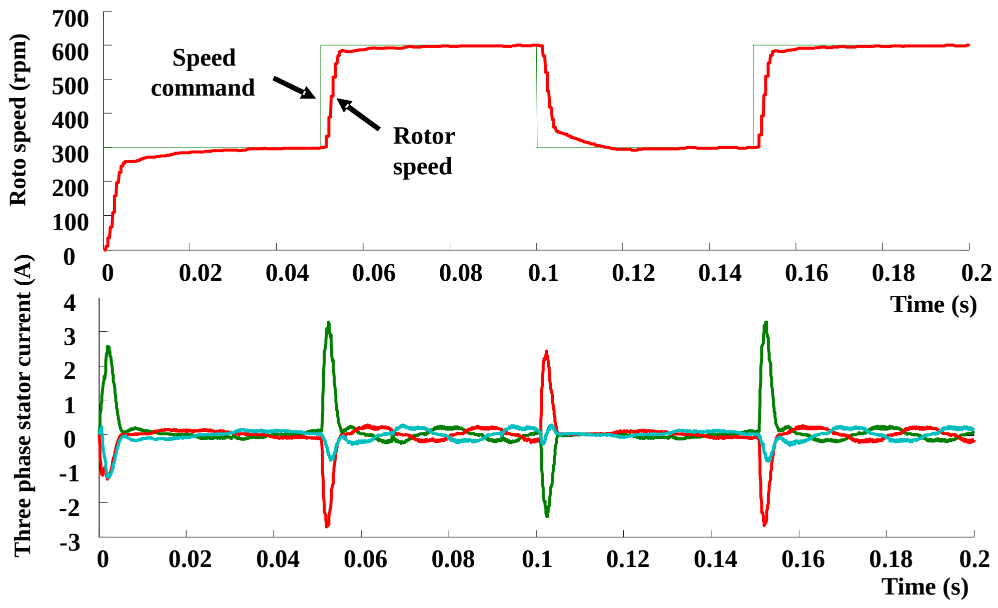


Figure 8. Simulation result when Fuzzy PI controller is used, and PMSM is operated at the normal-load condition.

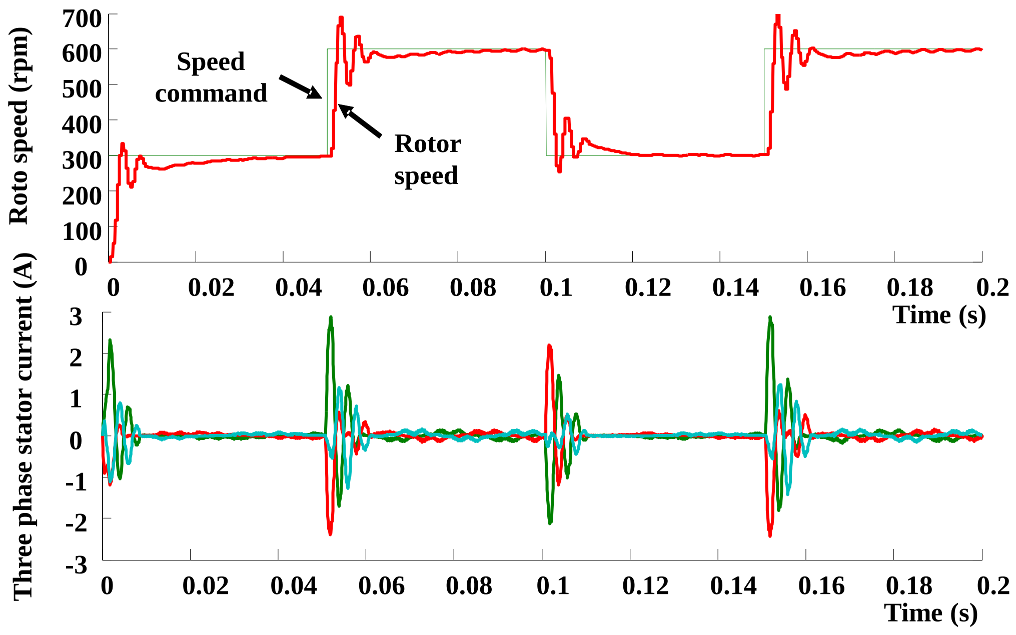


Figure 9. Simulation result when Fuzzy PI controller is used, and PMSM is operated at the light-load condition.

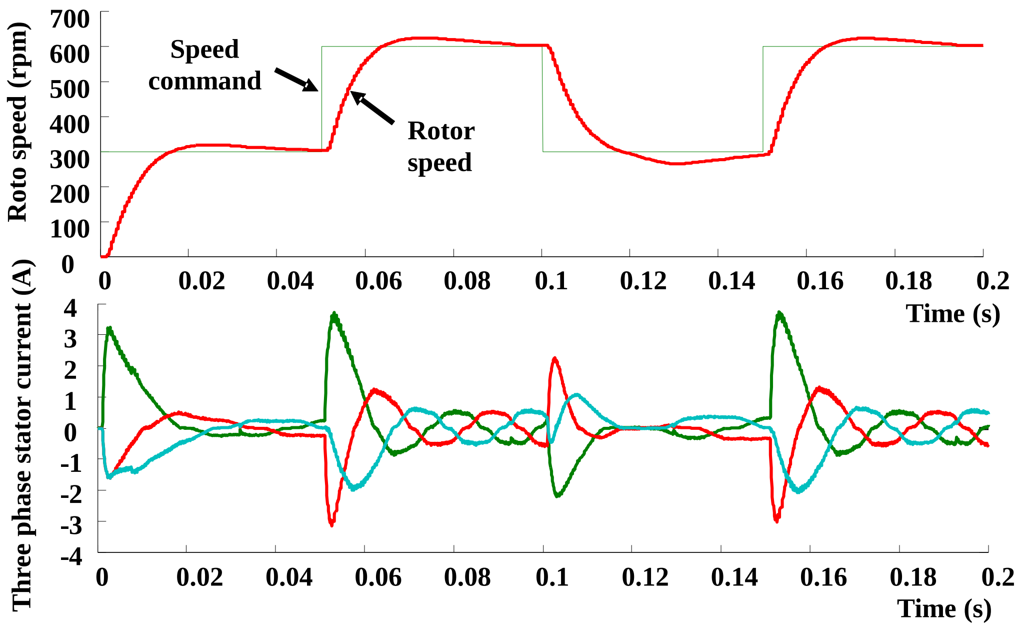


Figure 10. Simulation result when Fuzzy PI controller is used, and PMSM is operated at the heavy-load condition.

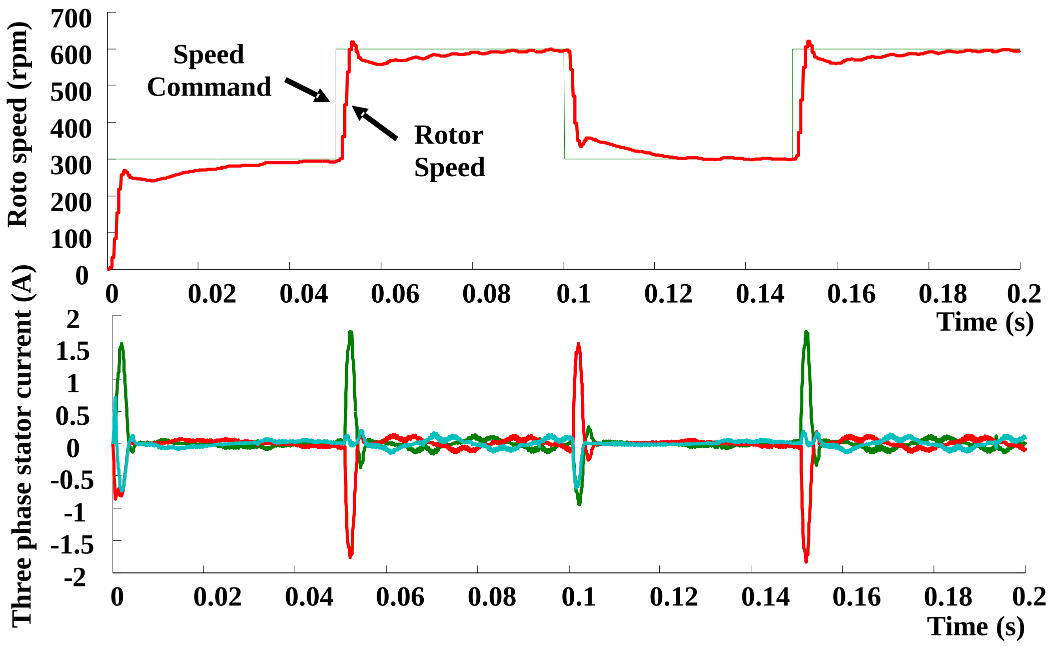


Figure 11. Simulation result when NFPC is used, and PMSM is operated at the light-load condition.

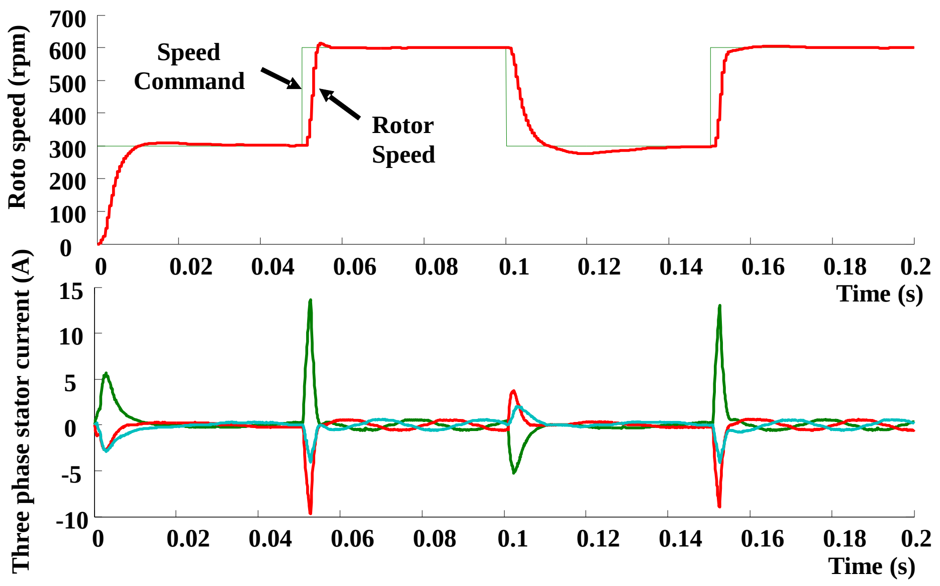


Figure 12. Simulation result when NFPC is used, and PMSM is operated at the heavy-load condition.

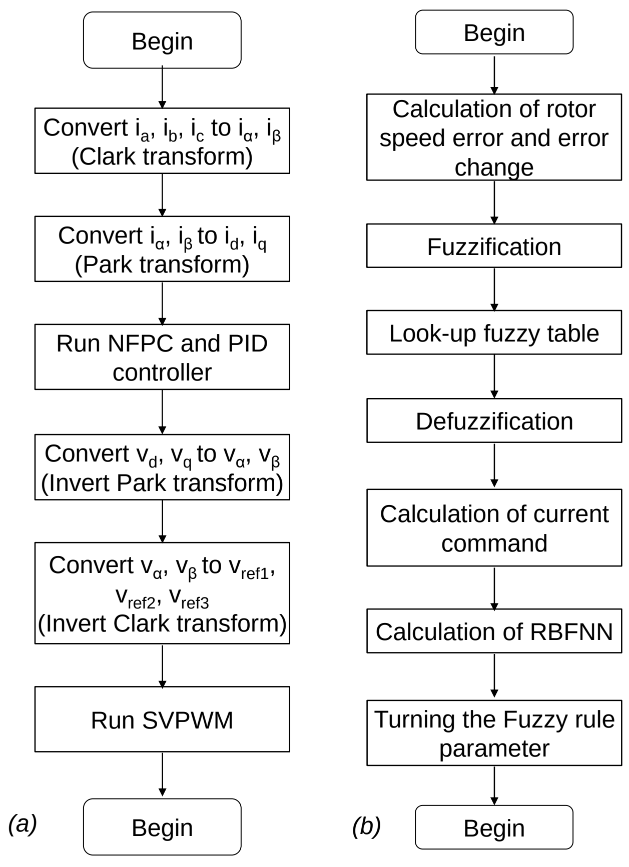

Begin
Convert i_a, i_b, i_c to i_α, i_β (Clark transform)
Convert i_α, i_β to i_d, i_q (Park transform)
Run NFPC and PID controller
Convert v_d, v_q to v_α, v_β (Invert Park transform)
Convert v_α, v_β to v_{ref1}, v_{ref2}, v_{ref3} (Invert Clark transform)
Run SVPWM
Begin
(a)
Begin
Calculation of rotor speed error and error change
Fuzzification
Look-up fuzzy table
Defuzzification
Calculation of current command
Calculation of RBFNN
Turning the Fuzzy rule parameter
Begin
(b)

Figure 13. The flowchart of the proposed algorithm for the PMSM speed controller. (a) the main program, (b) speed controller.

Acknowledgements. This work was supported by Lac Hong University, Viet Nam.

Declaration of competing interest. The author declares that he has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

1. Nguyen Doan Phuoc, Nguyen Hoai Nam - Settling time assignment with PID, Vietnam J. Sci. Technol. **55** (3) (2017) 357-367. DOI: 10.15625/2525-2518/55/3/8808
2. Quynh, N. V., Kung, Y. S., Van Dung, P., Liao, K. Y., & Chen, S. W. - FPGA-Realization of Vector Control for PMSM Drives, Applied Mechanics and Materials **311** (2013) 249-254.
3. Jung. J. W, Choi. Y. S, Leu. V. Q., Choi. H. H. - Fuzzy PI-type current controllers for permanent magnet synchronous motors, IET Electric Power Applications **5** (1) (2011) 143-152. DOI: 10.1049/iet-epa.2010.0036.
4. Quynh. N. V, Kung. Y. S - FPGA-realization of fuzzy speed controller for PMSM drives without position sensor," 2013 International Conference on Control, Automation and Information Sciences (ICCAIS), Nha Trang, (2013), 278-282.

   DOI: 10.1109/ICCAIS.2013.6720568
5. Kung. Y. S, Tsai. M.H - FPGA-based speed control IC for PMSM drive with adaptive fuzzy control, IEEE Transactions on Power Electronics **22** (6) (2007) 2476-2486. DOI: 10.1109/TPEL.2007.909185
6. Atif Iqbal, Haitham Abu-Rub & Hazem Nounou - Adaptive fuzzy logic-controlled surface mount permanent magnet synchronous motor drive, Systems Science & Control Engineering 2:1 (2014) 465-475. DOI: 10.1080/21642583.2014.915203
7. Li. S, Won. H, Fu. X, Fairbank. M, Wunsch. D. C, Alonso. E - Neural-Network Vector Controller for Permanent-Magnet Synchronous Motor Drives: Simulated and Hardware-Validated Results, in IEEE Transactions on Cybernetics **50** (7) 3218-3230 DOI: 10.1109/TCYB.2019.2897653
8. Zhang Chunmei, Liu Heping, Chen Shujin, Wang Fangjun - Application of neural networks for permanent magnet synchronous motor direct torque control, Journal of Systems Engineering and Electronics **19** (3) (2008) 555-561. DOI: 10.1016/S1004-4132(08)60120-6.
9. Le Hung Linh, Pham Thuong Cat, Pham Minh Tuan – Control of a three-phase AC motor with many uncertain parameters using a speed estimator, Journal of Computer Science and Cybernetics **29** (4) (2013) 313-324 (Vietnamese).
10. Nguyen Cat Ho, Vu Nhu Lan, Nguyen Tien Duy, Pham Van Them - Study the ability of replacing fuzzy and PI controllers with the hedge - algebra – based controller for a DC motor, Vietnam J. Sci. Technol. **52** (1) (2014) 35-48 (Vietnamese).
11. Zhong Y., Huang S., Luo D. - Stabilization and Speed Control of a Permanent Magnet Synchronous Motor with Dual-Rotating Rotors, Energies **11** (2018) 2786.
12. Strumiłło P., Kamiński W. - Radial Basis Function Neural Networks: Theory and Applications, Neural Networks and Soft Computing, Advances in Soft Computing, Physica, Heidelberg **19** (2003). DOI: 10.1007/978-3-7908-1902-1_14

13. Monmasson. E, Cirstea. M. N - FPGA design methodology for industrial control systems – a review, IEEE Trans. Ind. Electron. **54** (4) (2007) 1824-1842.
14. Tomasz Rudnicki - Measurement of the PMSM Current with a Current Transducer with DSP and FPGA, Energies **13** (1) 209 (2020); DOI: 10.3390/en13010209
15. Šmídl V., Nedvěd R., Košan T., Peroutka Z. - FPGA implementation of marginalized particle filter for sensorless control of PMSM drives, IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society, Vienna, 2013, pp. 8227-8232. DOI: 10.1109/IECON.2013.6700510
16. Mohammad Marufuzzaman, Mamun Bin Ibne Reaz, Labonnah Farzana Rahman and Tae Gyu Chang - FPGA Based Precise and High Speed Current dq PI Controller for FOC PMSM Drive, Current Nanoscience **10** (3) (2014) 394-401.

    DOI: 10.2174/1573413709666131203002033
17. Omar Sandre-Hernandez, Jose Rangel-Magdaleno, Roberto Morales-Caporal, E. Bonilla-Huerta - HIL simulation of the DTC for a three-level inverter fed a PMSM with neutral-point balancing control based on FPGA, Electr Eng. **100** (2018) 1441-1454. DOI: 10.1007/s00202-017-0597-0
18. Ying-Shieh Kung, Nguyen Vu Quynh, Nguyen Trung Hieu, and Jin-Mu Lin - FPGA Realization of Sensorless PMSM Speed Controller Based on Extended Kalman Filter, Mathematical Problems in Engineering, Volume **2013**, Article ID 919318, 13 pages, DOI: 10.1155/2013/919318
19. The Mathworks, Matlab/Simulink Users Guide, Application Program Interface Guide, 2004.
20. Modeltech, ModelSim Reference Manual, 2004.