# Design and Simulation of Adaptive Speed Control for SMO-Based Sensorless PMSM Drive

Ying-Shieh Kung [1], Nguyen Vu Quynh[2], Chung-Chun Huang[3] and Liang-Chiao Huang[4]

[1,2]Department of Electrical Engineering, Southern Taiwan University, Taiwan
[2]Department of Electrical Engineering, Lac Hong University, Vietnam
[3,4] Green Energy and Environment Research Laboratories, Industrial Technology Research Institute, Taiwan

***Abstract*- The work presents an adaptive PI controller for sensorless permanent synchronous motor (PMSM) drive system. A rotor flux position of PMSM is estimated by using a sliding mode observer (SMO), firstly. The estimated rotor position will send to the current loop for current vector control and simultaneously feedback to the speed loop for speed control. Then to increase the performance of the PMSM drive system, a PI controller which its parameters are tuned by a radial basis function neural network (RBF NN) is applied to the speed controller for coping with the effect of the system dynamic uncertainty. In realization, the Very high speed IC Hardware Description Language (VHDL) is adopted to describe the behavior of the sensorless speed control IP (Intellectual Property) which includes the circuits of space vector pulse width modulation (SVPWM), vector control, coordinate transformation, SMO, PI controller, RBF NN, etc. Further, a simulation work is performed by MATLAB/Simulink and ModelSim co-simulation mode, provided by Electronic Design Automation (EDA) Simulator Link. The PMSM, inverter and speed command are performed in Simulink as well as the sensorless speed control IP is executed in ModelSim. Finally, some co-simulation results validate the effectiveness of the proposed sensorless PMSM IP.**

## I. INTRODUCTION

Because the merits of high servo control performances and superior power density, PMSMs has been widely applied in the industrial automation machine as actuators. Nevertheless, the typical PMSM control needs a sensor to measure the rotor flux position and motor speed for ensuring the accuracy of current vector control and motor speed control, but it will relatively cause the problem of reliability and noise immunity. Therefore, in literature [1-7], the sensorless control for PMSM becomes a popular issue. Those sensorless control strategies have sliding mode observer, Kalman filter, neural network, etc. However, the back EMF and the sliding mode observer are suitable to be implemented by the fix-pointed processor and have been implemented in most studies. Further, in the industry applications, the PMSM driving system usually suffer from many uncertainties, such as model uncertainty, disturbance from external load, friction force, etc. which always diminish the performance quality of the pre-design specification. Although the PID controllers are widely used in the industrial process due to their simplicity and robustness [8], the fixed parameters can hardly adapt to uncertainty or time varying system [9]. To cope with this problem, many advanced control techniques, such as adaptive fuzzy control [10] and adaptive PID control [9] have been developed to obtain high control performance. In this paper, an adaptive PI controller based on RBF NN is adopted in speed loop of PMSM drive. The RBF NN is used to identify the plant dynamic and provided more accuracy plant information for parameters tuning of PI controller.

In recent year, An Electronic Design Automation (EDA) Simulator Link, which can provide a co-simulation interface between MALTAB/Simulink [11] and HDL simulators-ModelSim [12], has been developed and applied in the design of the motor drive and inverter system [13-16]. Using it you can verify a VHDL, Verilog, or mixed-language implementation against your Simulink model or MATLAB algorithm. In MATLAB/ Simulink environment, you can generate stimuli to ModelSim and analyze the simulation's responses [11]. In this paper, a co-simulation by EDA Simulator Link is applied to sensorless speed control for PMSM drive and shown in Fig.1. The PMSM, inverter and speed command are performed in Simulink and the sensorless estimation, the current vector control and the adaptive speed control IP described by VHDL code is executed in ModelSim. Some simulation results based on EDA Simulator Link demonstrate the correctness and effectiveness of the proposed sensorless PMSM IP in Fig.1.

## II. SYSTEM DESCRIPTION OF SENSORLESS PMSM DRIVE AND RBF NEURAL NETWORK CONTROLLER DESIGN

The sensorless speed control block diagram for PMSM drive is shown in Fig. 1. The modeling of PMSM, the SMO-based flux position estimation and the adaptive PI controller based on RBF NN identification are introduced as follows:

### *A. Mathematical Model of PMSM*

The typical mathematical model of a PMSM is described, in two-axis *d-q* synchronous rotating reference frame, as follows

$$\frac{di_d}{dt} = -\frac{r_s}{L_d} i_d + \omega_e \frac{L_q}{L_d} i_q + \frac{1}{L_d} \mathrm{v}_d \tag{1}$$

$$\frac{di_q}{dt} = -\omega_e \frac{L_d}{L_q} i_d - \frac{r_s}{L_q} i_q - \omega_e \frac{K_\mathrm{E}}{L_q} + \frac{1}{L_q} \mathrm{v}_q \tag{2}$$

where $\mathrm{v}_d$, $\mathrm{v}_q$ are the *d* and *q* axis voltages; $i_\mathrm{d}$, $i_\mathrm{q}$, are the *d* and *q* axis currents, $r_\mathrm{s}$ is the phase winding resistance; $L_\mathrm{d}$, $L_\mathrm{q}$ are the *d* and *q* axis inductance; $\omega_e$ is the rotating speed of magnet flux; $K_E$ is the permanent magnet flux linkage.

The current loop control of PMSM drive in Fig.1 is based on a vector control approach which will control the $i_\mathrm{d}$ to 0 and decouple the nonlinear model of PMSM to a linear system. Therefore, after decoupling, the torque of PMSM can be written as the following equation,

$$T_e = \frac{3P}{4} K_E i_q \triangleq K_t i_q \tag{3}$$

978-1-4577-1967-7/12/$26.00 ©2011 IEEE

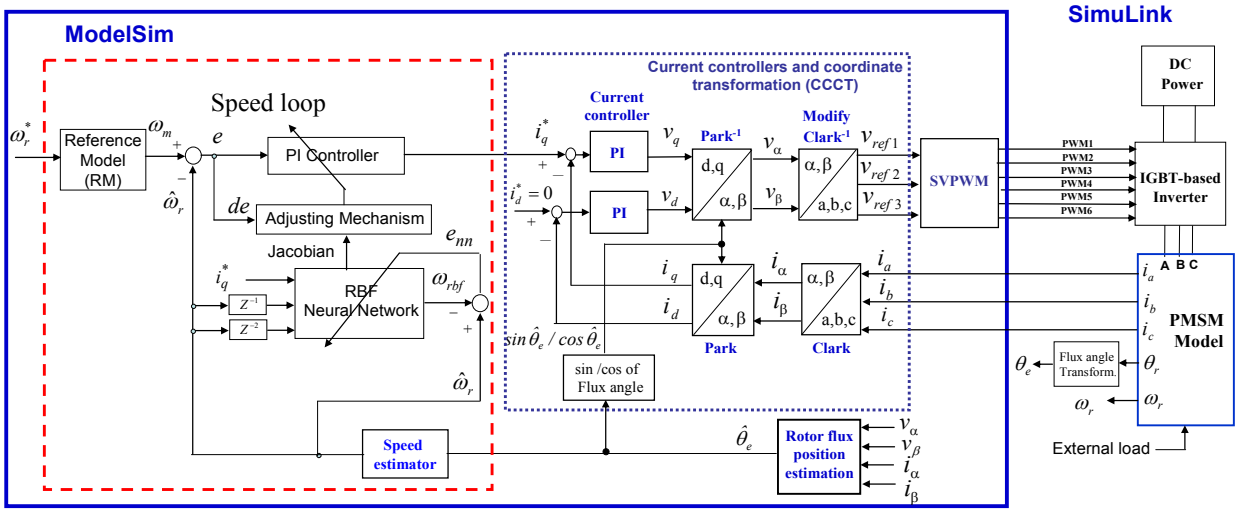


Fig.1 The block diagram of adaptive speed control for sensorless PMSM drive

Considering the mechanical load, the overall dynamic equation of PMSM drive system is obtained by

$$J_m \frac{d}{dt}\omega_r + B_m\omega_r = T_e - T_L \tag{4}$$

where $T_e$ is the motor torque, $P$ is pole pairs, $K_t$ is torque constant, $J_m$ is the inertial value, $B_m$ is damping ratio, $T_L$ is the external torque, $\omega_r$ is rotor speed.

*B. Algorithm of the rotor flux position estimation*

The block diagram to estimate the rotor flux position in Fig.1 is constructed in Fig. 2 which consists of a sliding mode observer (SMO), a bang-bang controller, a low-pass filter and a position computation. The inputs in this block are $i_\alpha(n), i_\beta(n), v_\alpha(n), v_\beta(n)$ , and the output is $\hat{\theta}_e$ . The algorithm of the rotor flux position estimation is presented as follows:

Step 1: Read the values of currents and voltages in $\alpha$ and $\beta$ axis, $i_\alpha(n), i_\beta(n), v_\alpha(n), v_\beta(n)$, from CCCT in Fig.1.

Step 2: Estimate the estimated current by SMO

$$\begin{bmatrix}\hat{i}_\alpha(n+1)\\ \hat{i}_\beta(n+1)\end{bmatrix} = \begin{bmatrix}\phi & 0\\ 0 & \phi\end{bmatrix}\begin{bmatrix}\hat{i}_\alpha(n)\\ \hat{i}_\beta(n)\end{bmatrix} + \psi\begin{bmatrix}v_\alpha(n)\\ v_\beta(n)\end{bmatrix} - \psi\begin{bmatrix}\hat{e}_\alpha(n)\\ \hat{e}_\beta(n)\end{bmatrix} \tag{5}$$

where $\phi \triangleq e^{-\frac{r_s}{L}T_s}$ , $\psi \triangleq \frac{1}{r_s}(1-e^{-\frac{r_s}{L}T_s})$ and $T_s$ is the sampling time.

Step 3: Calculate the current error by

$$\tilde{i}_\alpha(n) = \hat{i}_\alpha(n) - i_\alpha(n) \text{ and } \tilde{i}_\beta(n) = \hat{i}_\beta(n) - i_\beta(n) \tag{6}$$

Step 4: Obtain the Z gain of the current observer

$$Z(n) = \begin{bmatrix}z_\alpha(n)\\ z_\beta(n)\end{bmatrix} \triangleq k * sign\left(\begin{bmatrix}\tilde{i}_\alpha(n)\\ \tilde{i}_\beta(n)\end{bmatrix}\right) \tag{7}$$

Step 5: Estimate the EMF

$$\begin{bmatrix}\hat{e}_\alpha(n+1)\\ \hat{e}_\beta(n+1)\end{bmatrix} = \begin{bmatrix}\hat{e}_\alpha(n)\\ \hat{e}_\beta(n)\end{bmatrix} + 2\pi f_0\begin{bmatrix}z_\alpha(n)-\hat{e}_\alpha(n)\\ z_\beta(n)-\hat{e}_\beta(n)\end{bmatrix} \tag{8}$$

Step 6: Obtain the estimated rotor position

$$\hat{\theta}_e(n) = \tan^{-1}\left(-\frac{\hat{e}_\alpha(n)}{\hat{e}_\beta(n)}\right) \tag{9}$$

then set $n=n+1$ and back to Step 1.

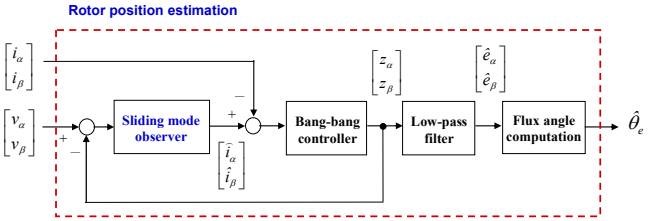


Fig.2 Rotor flux position estimation based on SMO

*C. Adaptive PI controller using RBF NN*

Adaptive PI controller in Fig.1 includes a PI controller, a reference model and a RBF NN for system identification. The detailed descriptions of those components are presented as follows.

(1) *PI controller*

In Fig. 1, digital PI controllers are presented in the speed loop of PMSM and the formulations are as follows.

$$e(k) = \omega_m(k) - \hat{\omega}_r(k) \tag{10}$$

$$u_p(k) = K_p\, e(k) \tag{11}$$

$$u_i(k) = K_i\sum_{j=2}^{k} e(j-1) = K_i\left(\sum_{j=2}^{k-1} e(j-1) + e(k-1)\right) \tag{12}$$

$$\triangleq K_i(E(k-2) + e(k-1)) = u_i(k-1) + K_i e(k-1)$$

$$i_q^*(k) = u_p(k) + u_i(k) = K_p e(k) + u_i(k-1) + K_i e(k-1) \tag{13}$$

with $E(k) = \sum_{j=2}^{k+1} e(j-1)$ and $u_i(k-1) = K_i E(k-2)$ . Besides, where $\hat{\omega}_r$ , $\omega_m$ , $e$ are the estimated rotor speed, the output of reference model and the error, respectively. The $K_p, K_i$ are $P$ controller gain and $I$ controller gain, respectively. The $u_p(k)$ , $u_i(k)$ , $i_q^*(k)$ are the output of P controller only, I controller only and the PI controller, respectively.

(2) *Radial basis function neural network (RBF NN)*

Fig. 3 shows the RBF NN which is three-layer architecture by an input layer, a single layer of nonlinear processing neurons and an output layer. The RBF NN has three inputs by $i_q^*(k)$, $\hat{\omega}_r(k-1)$, $\hat{\omega}_r(k-2)$ and its vector form is represented by

$$X = [i_q^*(k), \hat{\omega}_r(k-1), \hat{\omega}_r(k-2)]^T \tag{14}$$

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

$$h_r = \exp(-\frac{\|X - c_r\|^2}{2\sigma_r^2}), r = 1,2,3,4,....p \tag{15}$$

where $p$ is the number of neuron in hidden layer, $c_r = [c_{r1}, c_{r2}, c_{r3}]^T$ and $\sigma_r$ respectively denote 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 in Fig. 3 can be written as

$$\omega_{rbf} = \sum_{r=1}^{p} w_r h_r \tag{16}$$

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

Define the cost function as follows.

$$J = \frac{1}{2}(\omega_{rbf} - \hat{\omega}_r)^2 \triangleq \frac{1}{2} e_{nn}^2 \tag{17}$$

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

$$w_r(k+1) = w_r(k) + \eta e_{nn}(k) h_r(k) \tag{18}$$

$$c_{rs}(k+1) = c_{rs}(k) + \eta e_{nn}(k) w_r(k) h_r(k) \frac{X_s(k) - c_{rs}(k)}{\sigma_r^2(k)} \tag{19}$$

$$\sigma_r(k+1) = \sigma_r(k) + \eta e_{nn}(k) w_r(k) h_r(k) \frac{\|X(k) - c_r(k)\|^2}{\sigma_r^3(k)} \tag{20}$$

where *r=1,2,..p*, *s=1,2,3* and $\eta$ is a learning rate. Further, the Jacobian transformation can be derived from Fig.3 and (16) and it is

$$\frac{\partial \hat{\omega}_r}{\partial i_q^*} \approx \frac{\partial \omega_{rbf}}{\partial i_q^*} = \sum_{r=1}^{p} w_r h_r \frac{c_{r1} - i_q^*(k)}{\sigma_r^2} \tag{21}$$

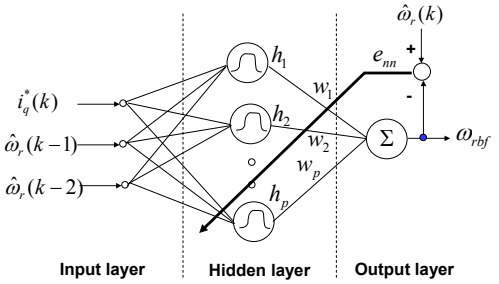


Fig.3 The architecture of RBF NN

(3) *Reference Model (RM):*

Second order system is usually considered to taken as the RM in the adaptive control system

$$\frac{\omega_m(s)}{\omega_r^*(s)} = \frac{\omega_n^2}{s^2 + 2\varsigma\omega_n s + \omega_n^2} \tag{22}$$

where $\omega_n$ is natural frequency and $\varsigma$ is damping ratio. Furthermore, applying the bilinear transformation, (22) can be transformed to a discrete model by

$$\frac{\omega_m(z^{-1})}{\omega_r^*(z^{-1})} = \frac{\theta_0 + \theta_1 z^{-1} + \theta_2 z^{-2}}{1 + \phi_1 z^{-1} + \phi_2 z^{-2}} \tag{23}$$

and the difference equation is written as.

$$\begin{aligned}\omega_m(k) = &-\phi_1\omega_m(k-1) - \phi_2\omega_m(k-2) + \theta_0\omega_r^*(k) \\ &+ \theta_1\omega_r^*(k-1) + \theta_2\omega_r^*(k-2)\end{aligned} \tag{24}$$

(4) *Adjusting Mechanism of PI Controller*

The gradient descent method is used to derive the tuning law of PI controller in Fig. 1. The adjusting mechanism is to minimize the square error between the rotor speed and the output of the reference model. The instantaneous cost function is firstly defined by

$$J_e \triangleq \frac{1}{2} e^2 = \frac{1}{2}(\omega_m - \omega_r)^2 \approx \frac{1}{2}(\omega_m - \hat{\omega}_r)^2 \tag{25}$$

and the parameters of PI controller are adjusted according to

$$\Delta K_p \propto -\frac{\partial J_e}{\partial K_p} = -\alpha \frac{\partial J_e}{\partial K_p} \tag{26}$$

And

$$\Delta K_i \propto -\frac{\partial J_e}{\partial K_i} = -\alpha \frac{\partial J_e}{\partial K_i} \tag{27}$$

where $\alpha$ represents learning rate. Secondly, the chain rule is used, and the partial differential equation for $J_e$ in (26) and (27) can be written as

$$\frac{\partial J_e}{\partial K_p} = \frac{\partial J_e}{\partial e}\frac{\partial e}{\partial \omega_r}\frac{\partial \omega_r}{\partial i_q^*}\frac{\partial i_q^*}{\partial K_p} \tag{28}$$

and

$$\frac{\partial J_e}{\partial K_i} = \frac{\partial J_e}{\partial e}\frac{\partial e}{\partial \omega_r}\frac{\partial \omega_r}{\partial i_q^*}\frac{\partial i_q^*}{\partial K_i} \tag{29}$$

Further, from (10), (13), (25) and $\omega_r \approx \hat{\omega}_r$, we can get

$$\frac{\partial J_e}{\partial e} = e \tag{30}$$

$$\frac{\partial e}{\partial \omega_r} \approx \frac{\partial e}{\partial \hat{\omega}_r} = -1 \tag{31}$$

$$\frac{\partial i_q^*(k)}{\partial K_p} = e(k) \tag{32}$$

$$\frac{\partial i_q^*(k)}{\partial K_i} = E(k-2) + e(k-1) = E(k-1) \tag{33}$$

Therefore, substituting (30)~(33) and (21) into (28) and (29), the parameters of PI controller in (26) and (27) can be adjusted by the following expression.

$$\Delta K_p(k) = \alpha e^2(k) \sum_{r=1}^{p} w_r h_r \frac{c_{r1} - i_q^*(k)}{\sigma_r^2} \tag{34}$$

$$\Delta K_i(k) = \alpha e(k) E(k-1) \sum_{r=1}^{p} w_r h_r \frac{c_{r1} - i_q^*(k)}{\sigma_r^2} \tag{35}$$

## III. SIMULINK/MODELSIM CO-SIMULATION OF SENSORLESS SPEED CONTROL FOR PMSM DRIVE

In Fig.1, it shows the sensorless speed control block diagram for PMSM drive and its Simulink/ModelSim co-simulation architecture is presented in Fig.4. The PMSM, IGBT-based inverter and speed command are performed in Simulink, and the sensorless speed controller described by VHDL code is executed in ModelSim with three works., The work-1 to work-3 of ModelSim in Fig.4 respectively performs the function of speed estimation and speed loop adaptive PI controller, the function of current controller and coordinate transformation (CCCT) and SVPWM, and the function of SMO-based rotor flux position estimation. The VHDL is used to describe the works in ModelSim. In current loop of PMSM drive, the sampling frequency is designed with 16kHz, and those in speed loop is 2kHz. The clocks with 20ns and 80ns periods are sent to work-1 and work3 of ModelSim.

A finite state machine (FSM) is employed to model the adaptive PI controller and SMO which are shown in Fig.5 and Fig.6, respectively. In Fig.5, it manipulates 81 steps machine to carry out the overall computations of an adaptive PI controller. The steps $s_0$~$s_5$ execute the reference model output; step $s_6$ perform the computation of speed error; steps $s_7$~$s_{10}$ execute the PI controller; steps $s_{11}$~$s_{74}$ describe the RBF NN and computation of Jacobian value and $s_{75}$~$s_{80}$ execute the PI gain tuning. The data format adopts 16-bit (Q15) with signed representation. The components of multiplier and adder use Altera LPM (Library Parameterized Modules) standard and its computation can be completed within 20ns. To prevent the numerical overflow condition occurred, the executing time at each step is designed with 80ns; therefore, in Fig.5, total 81 steps need 6.48μs. Further, In Fig.6, it manipulates 36 steps machine to carry out the overall computations. The steps $s_0$~$s_8$ execute the estimation of current value; steps $s_9$~$s_{10}$ compute the current error; $s_{11}$ is the bang-bang control; $s_{12}$~$s_{15}$ describe the computation of EMF and $s_{16}$~$s_{35}$ perform the computation of the rotor position. The data format adopts 12-bit (Q11) with signed representation. The components of multiplier and adder use Altera LPM standard but the component performing the arctan function is developed ourselves. The executing time at each step is designed with 80ns; therefore total 36 steps need 2.88μs. In Fig.4 the circuit design of CCCT and SVPWM in work-2 of ModelSim are not shown here. The FPGA (Altera) resource usages of work-1 to work-3 of ModelSim in Fig.4 are 8,942 LEs (Logic Elements) and 0RAM bits; 2,085 LEs and 24,576 RAM bits; 1,151LEs and 49,152 RAM bits, respectively.

## IV. CO-SIMULATION BASED ON EDA SIMULATOR LINK

Based on EDA simulator link, the simulation architecture for the proposed sensorless PMSM adaptive speed control system is presented in Fig.4. The ModelSim performs the function of adaptive PI controller, SMO and current vector controller which is described using VHDL code. In the Simulink, the SimPowerSystem blockset can provide the components of PMSM and the inverter and it also can generate stimuli to ModelSim and analyze the simulation's responses The designed PMSM parameters applied in simulation of Fig.4 are that pole pairs is 4, stator phase resistance is 1.3Ω, stator inductance is 6.3mH, inertia is J=0.000108 kg*m$^2$ and friction factor is F=0.0013 N*m*s.

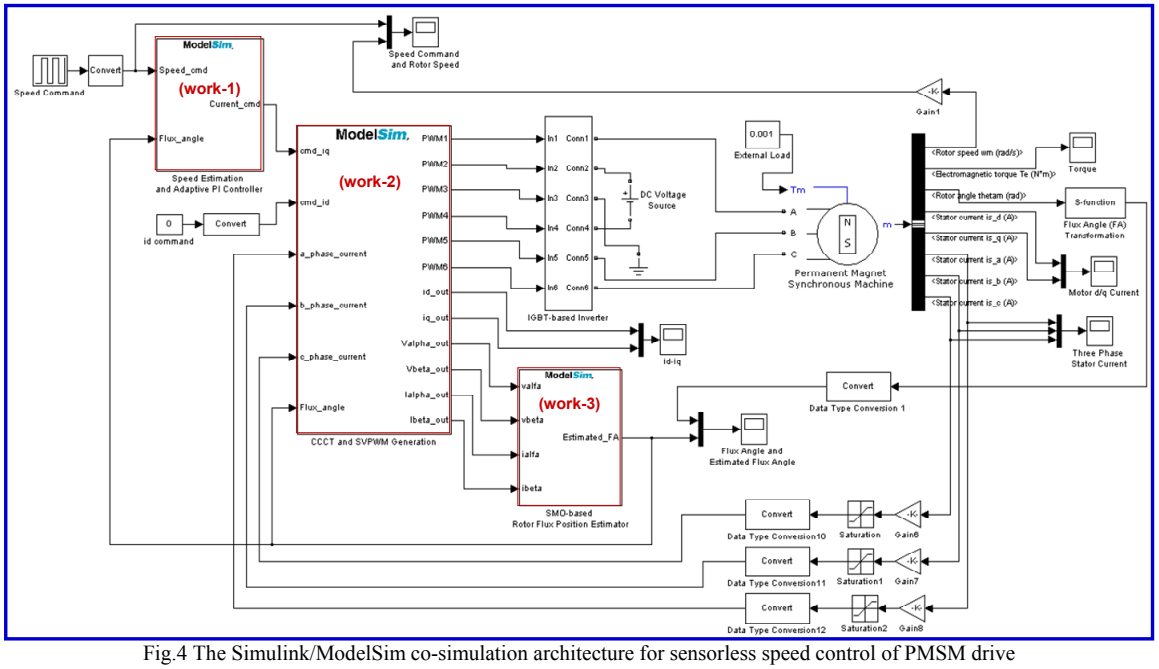


Fig.4 The Simulink/ModelSim co-simulation architecture for sensorless speed control of PMSM drive

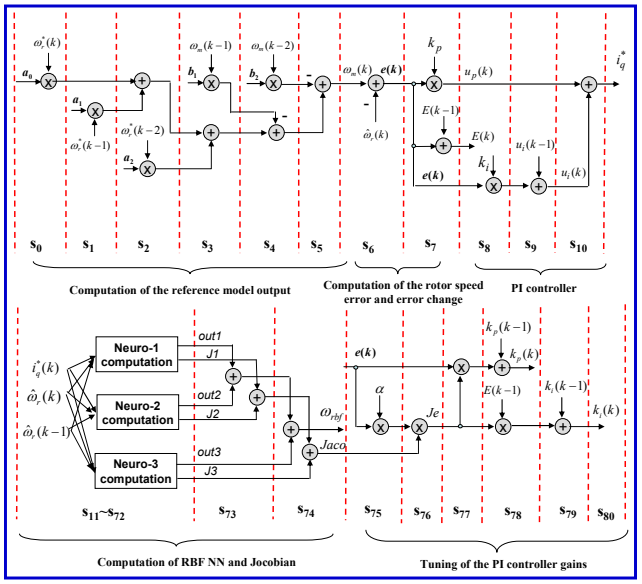


Fig. 5 State diagram of an FSM for describing the adaptive PI controller

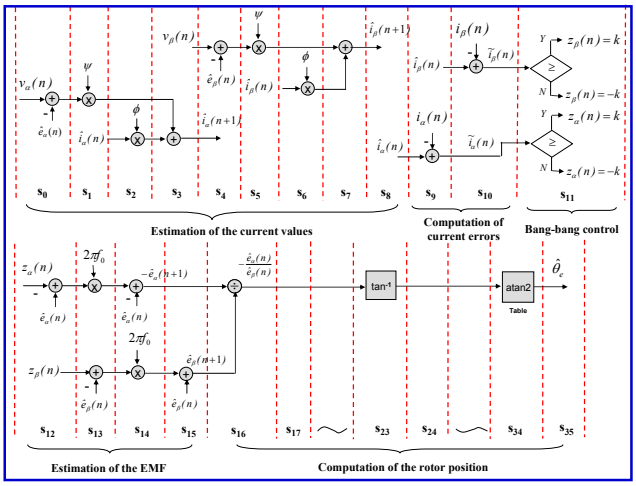


Fig.6 State diagram of an FSM for describing the SMO-based rotor position estimation algorithm

In the simulation of sensorless PMSM drive, rotor position estimation based on SMO is firstly evaluated. Three kinds of PMSM running speed at 500rpm, 1000rpm and 1500 rpm are tested and its simulation results of the real and estimated rotor flux position are presented in Fig.7. It shows that the response of the estimated rotor flux position $\hat{\theta}_e$ can follow with the actual rotor flux position $\theta_e$. Secondly, the performance of adaptive PI control using RBF NN identification is verified. Two tested cases are considered under different PMSM parameters, in which

Case 1: (Normal-load condition)

J=0.000108, F=0.0013 (36)

Case II: (Heavy-load condition)

J=0.000108*3, F=0.0013*3 (37)

When speed loop adopts PI controller only ($K_p$=1500, $K_i$=30) and sensorless PMSM drive runs at the normal-load condition and at 0~1500 rpm speed range, the simulation result of the step speed response with no overshoot and 0.25s rising time characteristics is shown in Fig.8. But when the running condition is changed to the heavy-load condition and speed range is operated from 0~800 rpm, the step speed response become worse with a little overshoot and sluggishness in Fig.9. It demonstrates that although the sensorless control based on SMO in PMSM drive can give a good speed tracking, it is still easily affected by external load variation. To cope with this problem, an adaptive PI control with RBF NN identification is adopted in Fig.1. The RBF NN will identify the plant dynamic and provide more accuracy plant information for parameters tuning of PI controller. Figures 10~11 show the simulation results while it uses the proposed adaptive PI control in sensorless PMSM drive. In this two Figs., the $K_p$ and $K_i$ are respectively set with 1500 (Q15 format) and 30 (Q15 format) at the initial condition; then $K_p$ and $K_i$ will be tuned to the adequate values to let the rotor response can follow the output of the reference model. Compare with Fig. 9, the result of Fig. 11 (c) shows an apparent improvement which the rotor speed can follow the output of RM after 1 sec. It also present that the proposed adaptive controller can enhance the robustness in sensorless PMSM drive.

## V. CONCLUSIONS

This study has been presented an adaptive speed control in SMO-based sensorless PMSM drive and successfully demonstrated its performance through co-simulation by using Simulink and ModelSim. In realization aspect, the VHDL is used to describe the behavior of the SMO estimator and the adaptive PI controller algorithm, and FSM method is applied to reduce the FPGA resource usage. In computational power aspect, the operation time to complete the computation of the SMO estimator and the adaptive PI controller algorithm are only 2.88μs and 6.48μs, respectively. In controller performance aspect, some simulation results show that the proposed adaptive PI controller for sensorless PMSM is effectiveness and robustness. After confirming the effective of VHDL code in adaptive PI control IP and rotor position estimation IP, the codes can be directly downloaded to FPGA for the use in sensorless PMSM drive.

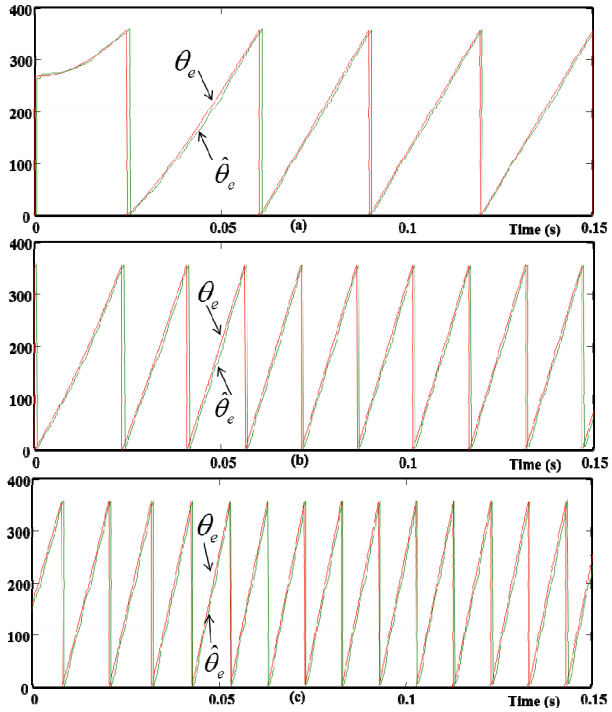


Fig. 7 Real rotor flux angle ($\theta_e$) and estimated rotor flux angle ($\hat{\theta}_e$) under PMSM speed running at (a)500rmp, (b)1000rpm and (c)1500rpm

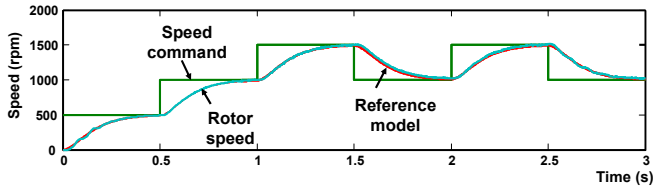


Fig. 8 Step speed response using PI controller only with $K_p$=1500, $K_i$=30 while sensorless PMSM operated at normal load condition

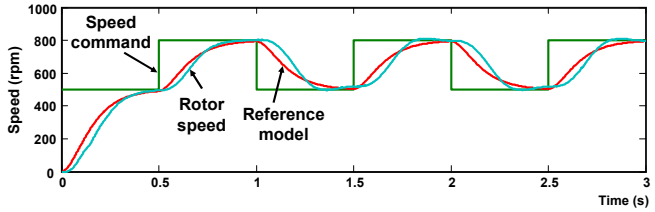


Fig. 9 Step speed response using PI controller only with $K_p$=1500, $K_i$=30 while sensorless PMSM operated at heavy load condition

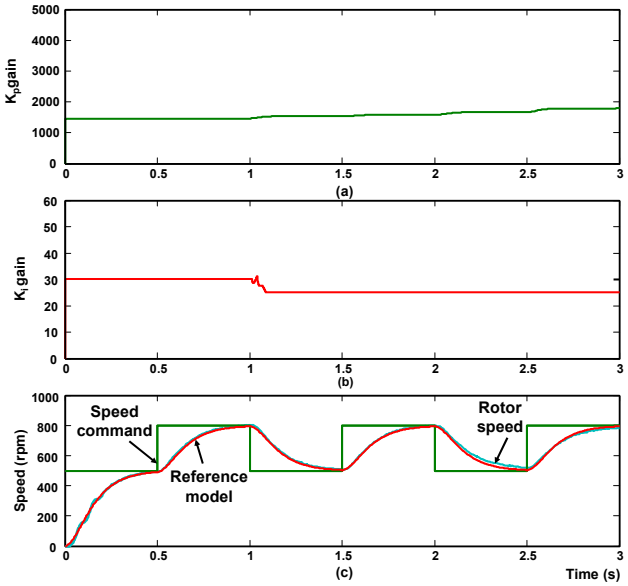


Fig. 10 Step speed response using adaptive PI controller while sensorless PMSM operated at normal load condition. (a) $K_p$ variation (b) $K_i$ variation (c) speed response

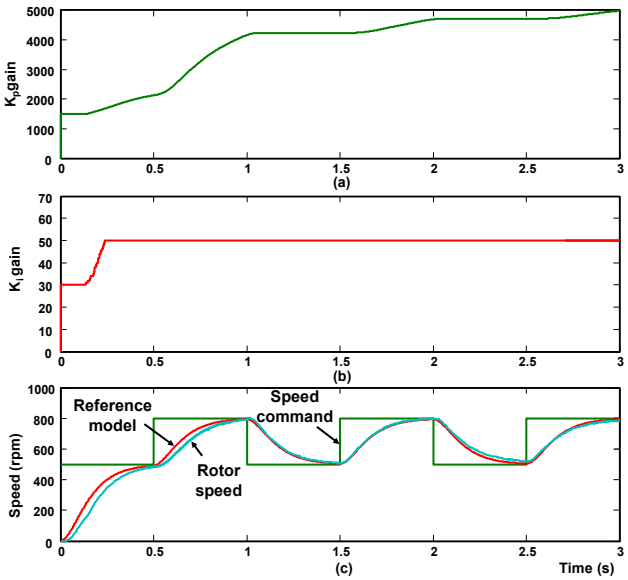


Fig 11. Step speed response using adaptive PI controller while sensorless PMSM operated at heavy load condition. (a) $K_p$ variation (b) $K_i$ variation (c) speed response.


## ACKNOWLEDGMENT

The financial support provided by Bureau of Energy is gratefully acknowledged.


## REFERENCE


[1] Y.S, Han, J.S. Choi and Y.S. Kim, "Sensorless PMSM Drive with Sliding Mode Control Based Adaptive Speed and Stator Resistance Estimator," *IEEE Trans. on Magnetics,* vol. 36, no. 5, pp.3588~3591, Sep. 2000.

[2] M. Ezzat, J.d. Leon, N. Gonzalez and A. Glumineau, "Sensorless Speed Control of Permanent Magnet Synchronous Motor by using Sliding Mode Observer," *in Proceedings of 2010 11th International Workshop on Variable Structure Systems*, pp.227~232, June 26 - 28, 2010.

[3] V.D. Colli, R.D. Stefano and F. Marignetti, "A System-on-Chip Sensorless Control for a Permanent-Magnet Synchronous Motor," *IEEE Trans. on Indus. Electron.*, vol. 57, no. 11, pp.3822~3829, Nov. 2010.

[4] T.D. Batzel and K.Y. Lee, "An Approach to Sensorless Operation of the Permanent-Magnet Synchronous Motor Using Diagonally Recurrent Neural Networks," *IEEE Trans. on Energy Conversion*, vol. 18, no. 1, pp.100~106, March 2003.

[5] P. Borsje, T.F Chan, Y.K. Wong, and S.L. Ho, "A Comparative Study of Kalman Filtering for Sensorless Control of a Permanent-Magnet Synchronous Motor Drive," *in Proceedings of IEEE International Conference on Electric Machines and Drives,* pp.815~822, 2005.

[6] M. C. Huang, A. J. Moses and F. Anayi and X. G. Yao, "Reduced-Order Linear Kalman Filter (RLKF) Theory in Application of Sensorless Control for Permanent Magnet Synchronous Motor(PMSM)," *in Proceedings of IEEE Conference on Industrial Electronics and Applications*, pp.1~6, 2006.

[7] L. Idkhajine and E. Monmasson, "Design methodology for complex FPGA-based controllers –Application to an EKF Sensorless AC Drive," *in Proceedings of International Conference on Electrical Machines (ICEM)*, pp. 1-6, 2010.

[8] K.J. Astrom, T. Hangglund, C.C. Hang and W,K, Ho, "Automatic Tuning and Adaptation for PID Controller – A survey," *IFAC J. Contr. Eng. Practice*, vol. 1, no.4, pp.699-714, 1993.

[9] M.G. Zhang, X.G. Wang and M.G. Liu, "Adaptive PID Control based on RBF neural network identification," in Proc. of the *IEEE International Conference on* Tools with Artificial Intelligence, Nov. 14-16, 2005.

[10] Y.S. Kung and M.H. Tsai, "FPGA-based Speed Control IC for PMSM Drive with Adaptive Fuzzy Control", *IEEE Trans. on Power Electronics*, vol. 22, no. 6, pp. 2476-2486, Nov. 2007.

[11] The Mathworks, Matlab/Simulink Users Guide, *Application Program Interface Guide*, 2004

[12] Modeltech, ModelSim Reference Manual, 2004.

[13] M.F. Tsai, Tran Phu Quy, B.F. Wu, and C.S. Tseng, "Model Construction and Verification of a BLDC Motor Using MATLAB/SIMULINK and FPGA Control," *in Proceedings of the 2011 6th IEEE Conference on Industrial Electronics and Applications,* pp.1797-1802, 2011.

[14] Y.S. Kung, V.Q. Nguyen, Chung-Chun Huang and L.C. Huang, "Simulink/ModelSim Co-Simulation of Sensorless PMSM Speed Controller," *in Proceedings of the 2011 IEEE Symposium on Industrial Electronics and Applications (ISIEA 2011)*, pp.24-29, 2011.

[15] D. Feng, L. Yan; L. Xu and W. Li "Implement of Digital PID Controller Based on FPGA and the System Co-simulation," *in Proceedings of the 2011 International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pp.921-924, 2011.

[16] M.F. Tsai, F. J. Ke, L.C. Hsiao, and J.K. Wang, "Design of a Digital Control IC of a Current Source Based on a Three-Phase Controlled Rectifier," *in Proceedings of the IEEE 6th International Power Electronics and Motion Control Conference (IPEMC*), pp.337-343, 2009.