Embedded Position Control of Permanent Magnet Synchronous Motor Using Model Predictive Control

Agus Ramelan[1], Arief Syaichu Rohman[1], and Allen Kelana[1]

[1]*School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Indonesia*

Abstract. This paper presents an implementation embedded system for position control of Permanent Magnet Synchronous Motor (PMSM). The control system consists of raspberry pi 3 as a microcontroller, ASDA-A2 servo drive, and Delta Servo ECMA type. The software design includes simulation tool and Python included on Raspbian OS. Communication between Raspberry Pi 3 and ASDA-A2 drivers using the ASCII Modbus communication protocol. Raspberry Pi 3 processes the reference data and the actual reading result and calculates the resulting error. The control algorithm used in this research is Model Predictive Control (MPC). As a Linear Quadratic Regulator, MPC aims to design and generate an optimal control signal with the ability to anticipate saturation, receding horizon, future event and take control accordingly In the design of the MPC technique to adjust the speed of the PMSM to take action of reference tracking, performance index optimization is done by adjusting the value of weighting horizon N, Q and R. The simulation and implementation results showed that the PMSM can reach the stability point on each desired setpoint and result in a near-zero steady-state error.

1 Introduction

Electric drives have an important role in the industry. One type of electric drive that is often used is a permanent magnet synchronous motor (PMSM)[1]. PMSM provides reliable speed control for acceleration and deceleration. As an actuator, there is often a saturation problem at the input. This will degrade system performance if it is not embedded in the design. With such non-linearity nature, it is difficult to obtain a stability guarantee [2]. For that, we need the method of MPC (Model Predictive Control) as one way to solve the problem. Example of implementation of MPC on drive system has been done before. That paper gives an exhaustive description of the design procedure of MPC applied to the combined control of the motor speed and current[3]. The implementation of an MPC on microcontroller also has been done. An example is the implementation of MPC on an Arduino and an FPGA board. MPC had implemented to control the speed of BLDC motor using Arduino Mega 2560. MPC gives better performance than PID and it is shown that MPC capable to handle input saturation of the system[4]. Besides that, there is a paper that discusses the MPC application for TITO (Two Input Two Output) system and its implementation onboard SPARTAN 6 FPGA SP605 Evaluation Kit with Microblaze MCS. The implemented MPC using Algorithm-3 on board can be used by a different plant with fast responses[5]. The differences that the author did with the previous one, will be discussed at the end of this chapter. MPC is one of the control methods that calculate the current control input by solving an optimal control problem of a limited open-loop horizontal and using the first element of the optimum control sequence at each sampling time. This method has the ability to overcome saturation on both the control input, system input, and system output. The saturation element is included as an additional requirement that must be calculated in resolving the problem of optimizing the objective function of MPC continuously. The process of optimizing quadratic objective function with restrictions on its variables is called quadratic programming (QP). An algebraic loop that involves saturation equivalent to a QP problem that has its variable constraints. Iterative of QP with an algebraic loop which produces MPC convergent solution may become slow, so we need a fast QP iteration algorithm for tracking such as Algorithm-2 and Algorithm-3[6].

In this paper, we will discuss the performance of MPC by using the algorithms-2[6] which then evaluated its performance. The new thing in this study is the microcontroller, motor type, driver, and communication system used. The system used in the implementation is a position control system PMSM controlled by Raspberry Pi 3. Type of motor used is ECMA K11820SS with its driver type ASDA-A2-2 43-M. The communication series used is RS-232 Modbus, where ASDA driver as master and Raspberry Pi 3 as the slave.

2 System Design

The PMSM motor control system model used is PMSM 2 KW 3 Phase and Driver Motor with 24 Volt supply. Mathematical modeling of PMSM has been done by previous research, for example in the paper [7]. In this study, the modeling is done by identification system tool. This study was conducted as one of the stages of rail simulator making. It consists of four integrated controlled PMSM motors. Before real implementation, a controller test is required for a quarter of the model. We use raspberry pi because it has GPIO port, HDMI, and USB. So it will be easier in programming and interconnect with other devices.

The ASDA-A2 driver is a PMSM motor controller driver which is a production of Delta Company. In this driver is equipped with features that are complete enough for the purposes of motor control. These features are input-output channels, communication ports with master/slave controllers and other devices, encoder sensors, writable and readable registers with other devices, and complete built-in HMI software [8]. Figure 1 is a block diagram of the PMSM motor position control system module with Raspberry Pi 3 based Raspberry Algorithm.

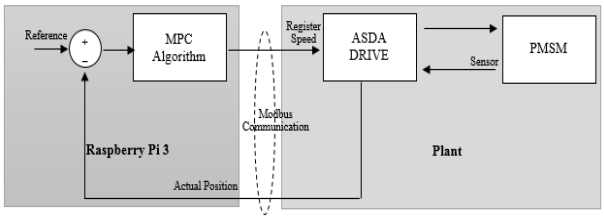

Figure 1. Block diagram of system design.

Raspberry Pi 3 processes the reference data and the actual reading result and calculates the resulting error. This error is the input for predictive model control. The result of predictive model control calculation is a control signal in the form of the input value of speed to be given to actuator that is PMSM motor. In the implementation of this control signal is limited in accordance with the specificity of the minimal-maximal values.

3 Controller Design

3.1 MPC Design

MPC or also known as receding horizon control is a control technique in the form of discrete time and has a purpose to predict the future behavior of the system process to obtain optimal closed-loop control that minimizes an objective function on-line to a certain time horizon with certain limits [9]. Broadly speaking, the controlling stages of MPC, in general, can be explained as follows,

a. Calculate the output to come for predefined horizons,
b. Calculates the control input sequence that will come by optimizing the objective function,
c. From the set of control signals, only the first *u(k)* input is assigned to the system and back to step a.

In order to achieve zero tracking errors for a position control, adding integrator is not needed. The open loop system can be written as the following equation.

$$x_{k+1} = A_d x_k + B_d u_k \quad (1)$$

$$y_k = C_d x_k + D_d u_k \quad (2)$$

The problem with the MPC strategy is to design a control by finding U, within the constraint of U_{sat}, that minimizes the objective function J along the horizon length, N, at each sampling time.

$$J = e_{k+N}^T \overline{Q}_N e_{k+N} + \sum_{j=0}^{N-1} \left(e_{k+j}^T \overline{Q} e_{k+j} + u_{k+j}^T \overline{R} u_{k+j} \right) \quad (3)$$

$$J = e_k^T \overline{Q} e_k + E^T Q E + U^T R U \quad (4)$$

$$J = J_x + X^T \left(\overline{\Phi}^T Q \overline{\Phi} \right) X + U^T M U + 2 U^T F_X X \quad (5)$$

where

$$\overline{Q} = \overline{Q}^T > 0, \overline{R} = \overline{R}^T > 0 \quad (6)$$

$$X = \begin{bmatrix} x_k \\ -\Omega \end{bmatrix}; Y = \begin{bmatrix} y_{k+1} \\ y_{k+2} \\ \vdots \\ y_{k+N} \end{bmatrix}; \Omega = \begin{bmatrix} \omega_{k+1} \\ \omega_{k+2} \\ \vdots \\ \omega_{k+N} \end{bmatrix}; E = \begin{bmatrix} u_{k+1} \\ u_{k+2} \\ \vdots \\ u_{k+N} \end{bmatrix} \quad (7)$$

minimizing J is equivalent to minimizing J_R within the constraint of U_{sat}.

$$\frac{1}{2}(U + M^{-1} F_X X)^T M (U + M^{-1} F_X X) = J_R \quad (8)$$

According to [6], by comparing equation [8] and [13]

$$-U_0 = M^{-1} F_k x_k \quad (9)$$

$$U_0 = M^{-1} F_k x_k \quad (10)$$

The objective function optimization algorithm used is QP and can be solved by a class of algebra loop [6]. The results stated that a class of algebra loop could be used to complete an on-line QP. One of the block diagrams which is an alternative QP solution can be explained on the Figure 2.

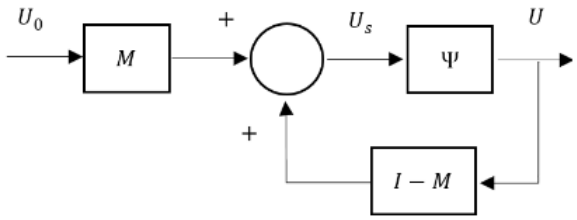

Figure 2. Nonlinear algebraic loop.

The loop algebra shown in Figure 2 is equivalent to solving the following QP problem based on the objective function:

$$U = \arg\min(J), U \in U_{sat} \quad (11)$$

$$U_{sat} = U_{min}, \leq U_i \leq U_{max}, , i = 1,2, \dots N \quad (12)$$

and

$$J = \frac{1}{2}(\mathrm{U} - \mathrm{U0})^T M(U - U0) \quad (13)$$

$$M = M^T > 0 \quad (14)$$

$$\overline{\Phi} = [\Phi \quad I_N]; \overline{\Gamma} = \Gamma; \quad (15)$$

$$\overline{\Phi} = \begin{bmatrix} C_d A_d \\ C_d {A_d}^2 \\ \vdots \\ C_d {A_d}^n \end{bmatrix};$$

$$\Gamma = \begin{matrix} C_d B_d & 0 & \dots & 0 \\ C_d A_d B_d & C_d B_d & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_d {A_d}^{n-1} B_d & C_d {A_d}^{n-2} B_d & \dots & C_d B_d \end{matrix}; \quad (16)$$

Finally, the first u_k control vector in U is used as the current control input for the control.

$$u_k = [I_m \, 0 \dots 0]U \quad (17)$$

$$u_k = E_1^T U \quad (18)$$

3.2 Modelling

3.2.1 Plant Model

Modeling is done with the Identification System tool. The transfer function is obtained by giving varying inputs to the system. Input and output are the reference values of speed and actual read speed. The transfer function is approximated by order 2 with 1 zero in the identification system tool. The result of the transfer function is then integral to get the transfer function in position. From the modeling results obtained the function of continuous transfer of PMSM motor system is as follows.

$$G(s) = \frac{653.5\, s + 15600}{s(s^2 + 266.6\, s + 15600)} \quad (19)$$

Sampling time Ts is chosen with the longest computation time in the input stage is 0.03. Ts is chosen based on the consideration of the average speed of data transmission and command execution is 25 ms. So we get the transfer function in the form of discrete as follows.

$$G(z) = \frac{0.05061\, z^2 - 0.02072\, z - 0.00224}{z^3 - 1.079\, z^2 + 0.07901\, z - 0.0003361} \quad (20)$$

The model of the control must be detectable and stabilizable. One more step in the prediction can be used recursively to find the next prediction. The prediction can be specified in length based on the horizon selection. The output of the MPC is expected to follow a reference input $\omega_k \in \mathbb{R}^n$.

3.2.2 Implementation

Implementation of MPC consists of several steps, such as calculate state estimation, calculate the output to come for predefined horizons, calculates the control input sequence that will come by optimizing the objective function. Finally, from the set of control signals, only the first *u(k)* input is assigned to the system and back to the first step. The block diagram of MPC implementation can be explained on the figure 3.

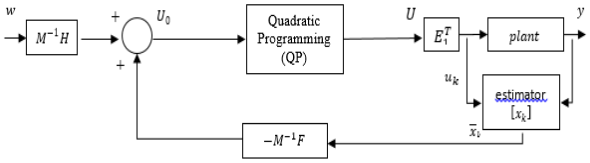

Figure 3. Block diagram of MPC implementation.

3.2.3 Quadratic Programming

The quadratic programming to be used is algorithm-2 which is quadratic iteration programming algorithm with block diagram as figure 4.

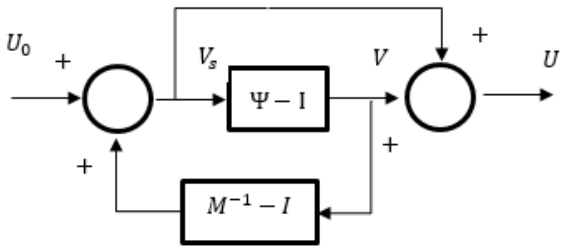

Figure 4. Algorithm-2 block diagram.

$$\Psi_a = \Psi_a - I \quad (21)$$

Ψ is diagonal saturation that representing U_{sat}. The loop equation will be as follows:

$$V = \Psi_a(V_s) = \Psi_a(U_0 + (M^{-1} - I)V) \quad (22)$$

$$U = V_s + V = M^{-1}V + U_0 \quad (23)$$

That iteratively becomes

$$V_{k+1} = \Psi_a(U_0 + (M^{-1} - I)V_k) \quad (24)$$

If iterating convergent, then the solution U^* is obtained as follows.

$$U^* = V^* + (M^{-1} - I)V^* + U_0 = M^{-1}V^* + U_0 \quad (25)$$

3.2.4 State Estimator

Given that the built model is not known the relationship in detail through the mathematical model, and the model obtained is done with tools identification system. Then all state is estimated by Luenberger approach of full order state estimation as shown in figure 5.

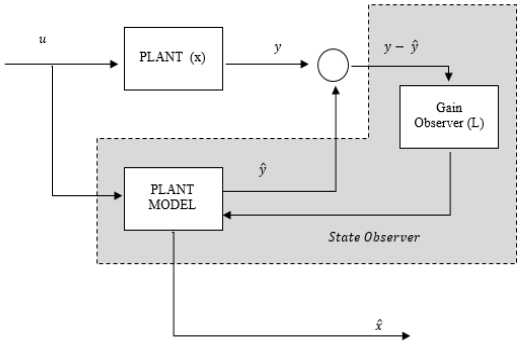

Figure 5. Algorithm-2 block diagram.

The equation of full order state estimator (observer) as shown above is as follows

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L(y(k)-\hat{y}(k)) \quad (26)$$

$$\hat{Y}(k) = C\hat{x}(k) + Du(k) \quad (27)$$

4 Simulation and Experimental Result

Both in simulation and implementation, the selection of the same parameters. These parameters include the time sampling, the horizon, Q, R, and the length of the iteration. In this study, selected time sampling = 0.03 s, N = 33, Q = 185, R = 1, and the length of iteration. The horizon is chosen with considering the desired closed-loop response time is T and the control interval is Ts, try p such that T ≈ pTs[10]. The QP iteration length is not limited to convergent because each input signal requires a different length of iteration.The simulation is done with simulation software and the implementation is done with Python programming language.

4.1 Simulations Result

In this simulation, the horizon value is chosen based on the reference to the MATLAB library[10] where the horizon is the comparison between the targeted time response (1 s) with the sampling time (0.03 s). Other parameters are selected by trial to produce the best results. In the first simulation is done rounding the nearest value of control signal according to the requirement on ASDA system that requires integer value. MPC requires the state value of the system to perform computations but only known y (k) system outputs. From this it is estimated the state value with full order state estimator. Gain estimator L can be searched by Pole Placement (Ackerman). The iteration termination criterion is performed with a convergent error <1%. The simulation results are shown in the following figure 6 and figure 7.

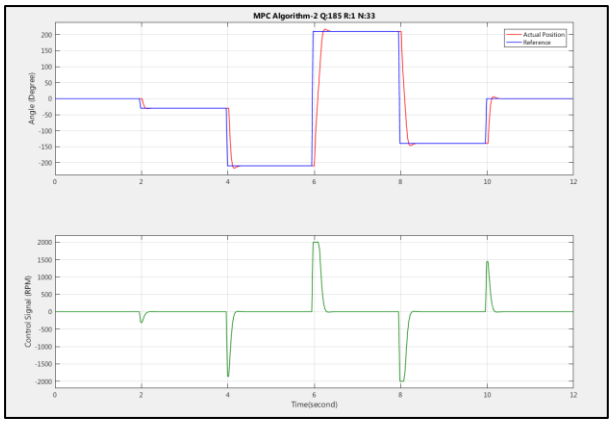

Figure 6. Respone system for tracking trajectory.

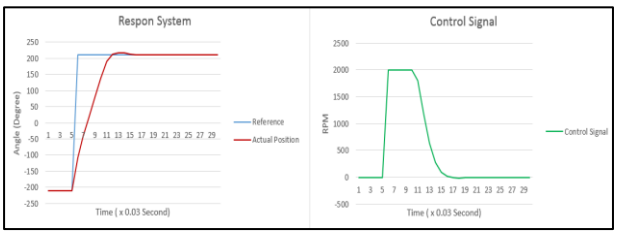

Figure 7. Respone System for Tracking Trajectory at 6 seconds ≤ t ≤ 8 seconds.

Settling time and steady state error are reviewed from the reference input and output when the output is steady at 6 seconds ≤ t ≤ 8 seconds, ie -210º to 210º. From the results it can be seen that the settling timenya is 0.45 seconds and the steady state error is 0.03 degree. So the simulation results can be concluded that the system response meets the desired specification of settling time less than 1 second and steady state error close to equal 0 degree.

4.2 Experimental Result

Similarly, with simulations, experiments are performed by giving variable reference inputs in a clockwise and counterclockwise direction from the shortest to the farthest angle. The farthest reference value is -210º to 210º. The experimental results are shown in the following figure 9 and figure 10.

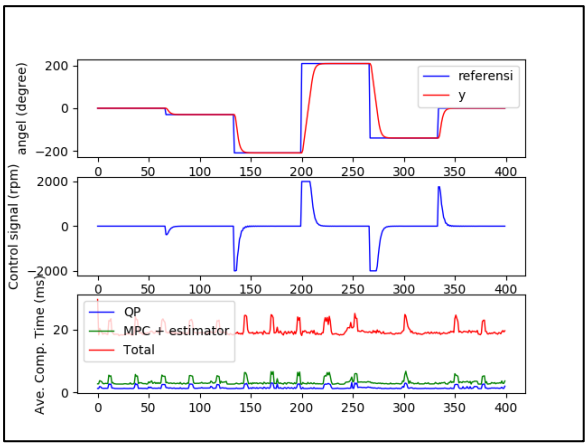

Figure 8. Response system for tracking trajectory.

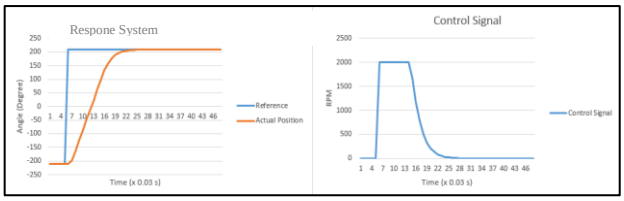

Figure 10. Respone system for tracking trajectory at 6 seconds ≤ t ≤ 8 seconds.

Settling time and steady state error are reviewed from the reference input and output when the output is steady at 6 seconds ≤ t ≤ 8 seconds, ie -210º to 210º. From the results it can be seen that the settling timenya is 0.66 seconds and the steady state error is 0.0096 degree. In addition, Figure 9 also shows the average calculation time QP = 1.62 ms, MPC + Estimator = 3.49 ms, total time for execution 1 signal reference = 20.1050 ms, and the maximal of total time for execution 1 signal reference = 28.31 ms. So the experimental results can be concluded that the system response meets the desired specification of settling time less than 1 second and steady state error close to equal 0 degree.

5 Conclusion

The Model Predictive Control has been successfully simulated and implemented as a Permanent Magnet Synchronous Motor (PMSM) position controller. MPC is computed inside Raspberry Pi 3 and the resulting control signal is configured with ASDA-Servo with serial communication rs-232 Modbus. The use of raspberry memory for this MPC implementation is about 25%. The simulation and experimental results can be concluded that the system response meets the desired specification of settling time less than 1 second and steady state error close to equal 0 degree. Their results show similarities. In the simulation occurs a small overshot while the implementation is not. However, this difference is not so significant.

6 Acknowledgments

This work was supported by LPDP Scholarship from Indonesian Ministry of Finance, School of Electrical Engineering and Informatics, Institut Teknologi Bandung, especially Control and Intelligence System Research Group.

References

1. F. De Lillo, F. Cecconi, G. Lacorata, A. Vulpiani, EPL, 84 (2008) [1] Aamir Hashim O.A. “Optimal Speed Control for Direct Current Motors Using Linear Quadratic Regulator”. Journal of Science and Technology - Engineering and Computer Sciences, Vol. 14, No. 3, June 2013.
2. Prasetyo, Hanif. (2017): Quadratic Programming Implementation in Model Predictive Control using Arduino Mega2560 for Speed Control of BLDC Motor , Master’s Program Thesis, Institut Teknologi Bandung.
3. Saverio Bolognani, Silverio Bolognani, Luca Peretti, and Mauro Zigliotto. Design and Implementation of Model Predictive Control for Electrical Motor Drives. IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 56, NO. 6, JUNE 2009.
4. Hanif Fauzan Prasetyo; Arief Syaichu Rohman; M. R. A. R. Santabudi. Implementation of model predictive control using Algorithm-3 on Arduino Mega 2560 for speed control of BLDC motor. 2017 3rd International Conference on Science in Information Technology (ICSITech).
5. Cahyantari Ekaputri; Arief Syaichu-Rohman. Model predictive control (MPC) design and implementation using algorithm-3 on board SPARTAN 6 FPGA SP605 evaluation kit. 2013 3rd International Conference on Instrumentation Control and Automation (ICA).
6. Syaichu-Rohman, Arief, R.H. Middleton, and M.M Seron. “A Multivariable Nonlinear Algebraic Loop as a QP with Application to MPC.” The Proceedings of the European Control Conference. Cambridge, 2003.
7. A. B. Dehkordi, A. M. Gole, and T. L. Maguire. “Permanent Magnet Synchronous Machine Model for Real- Time Simulation”. International Conference on Power Systems Transients (IPST’05) in Montreal, Canada on June 19-23, 2005 Paper No. IPST05 - 159.
8. Delta Electronics Inc. “Delta High Resolution AC Servo Drive for Network Communication Applications ASDA-A2 Series”. User Manual. 2017.
9. Dede Irawan Saputra. “Electric Speed Control In Electric Vehicle Simulator Using Model Predictive Control”. Thesis Magister Electrical Engineering STEI Bandung Institute of Technology. 2017
10. MATLAB R2017b Help. Avaible on : https://www.mathworks.com/help/mpc/ug/choosing-sample-time-and-horizons.html