IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 27, 2019, accepted March 15, 2019, date of publication March 25, 2019, date of current version April 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2907111

Embedded Model Predictive Control for Enhancing Tracking Performance of a Ball-and-Plate System

HEESEUNG BANG AND YOUNG SAM LEE
Department of Electrical Engineering, Inha University, Incheon 22212, South Korea
Corresponding author: Young Sam Lee (lys@inha.ac.kr)

This work was supported by the Inha University Research Grant.

ABSTRACT Implementing model predictive control (MPC) on a limited-performance microcontroller has always been a great challenge. In this paper, we propose an MPC implementation method that can run on a microcontroller by improving the calculation efficiency and address applied to the ball-and-plate system to enhance tracking performance. First, we convert the MPC quadratic programming problem, which causes difficulties in implementing MPC, into an equivalent nonnegative least-squares problem so that we can build the solver in a C program easily. Subsequently, we separate the offline calculation from the online process, which drastically reduces the calculation time. Finally, we implement the proposed MPC in a C program to run it on a Nucleo-32 microcontroller, apply it to a tracking control problem using a laboratory-built ball-and-plate system, and then explain the improved tracking performance compared with the conventional control methods.

INDEX TERMS Model predictive control, quadratic programming, nonnegative least-squares, microcontroller, ball-and-plate system.

I. INTRODUCTION

In recent decades, ball-and-plate systems have been widely used as a basic educational tool for those who study control theory [1], [2]. Because the ball-and-plate system is an unstable, nonlinear system, it is effective for verifying various types of controllers and their performances and also helping students to see and understand the control process. Consequently, considerable research has been conducted on linear quadratic (LQ) control [3], sliding mode control (SMC) [1], [4], fuzzy control [5], [6], and proportional-integral-derivative neural-network control [7]. In general, the purpose of the control of the ball-and-plate system is stabilization or trajectory tracking. Stabilization means keeping the ball on the center of the plate, and trajectory tracking entails following a time-varying reference trajectory to draw a certain shape on the plate, such as a circle or a rectangle. However, the tracking performances found in [1]–[6] vary significantly depending on the reference because the controller cannot consider the time-varying reference. The research in [8] used model predictive control (MPC) to solve this problem, but it has only been verified via computer simulation.

MPC is a control technique that solves the optimization problem at each sampling time. MPC provides a systematic approach to handle physical constraints and reduces the tuning effort for new applications [9]. However, the commercial application of MPC on embedded systems is not widespread because the MPC optimization problem is a quadratic programming (QP) problem, which has intense computational demands. This has led to MPC not being used for systems that require fast sampling times or have a limitation on memory resources. Although there has been significant interest in applying MPC to robot systems (e.g., [10]–[12]), such applications still require a high-performance processor. Another approach is to use a lookup table that is generated through offline calculation. This approach helps to reduce online calculation time, but it requires a huge amount of memory.

In this paper, we propose an implementation method for embedding MPC on a microcontroller to enhance tracking performance of the ball-and-plate system. The main idea is to convert the QP problem into a least-distance problem (LDP) and then recast it to a nonnegative least-squares (NNLS) problem. The proposed method is effective

The associate editor coordinating the review of this manuscript and approving it for publication was Bohui Wang.

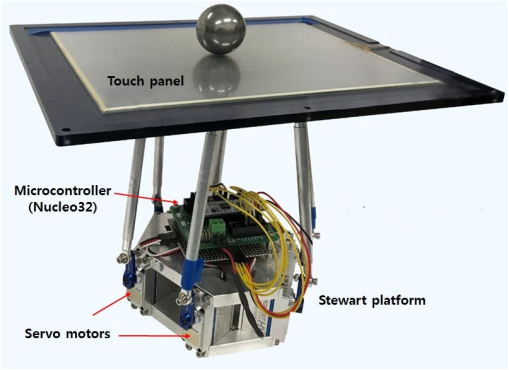


FIGURE 1. Photograph of the laboratory-built ball-and-plate system.

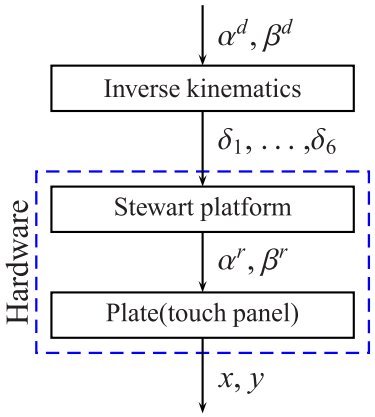


FIGURE 2. Process of rotating the plate.

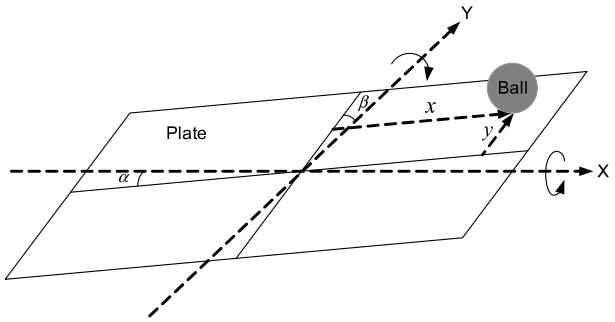


FIGURE 3. Diagram of the ball-and-plate system.

because constructing a QP solver within a C program is easy. The C-code application of the NNLS problem is widespread and it is even available on a website [13]. We build our own QP solver using the existing NNLS solver and then verify the calculation time and the quality of the tracking performance for a sinusoidal trajectory by comparing our results with those of other controllers such as sliding mode and linear LQ controllers. In this study, we use the laboratory-built ball-and-plate system (see Fig. 1) that was used in [1] and the microcontroller board is a Nucleo-32 board, which is a low-cost open-source piece of hardware. By using the proposed method, the system can handle the online calculation process in real time, thus enabling MPC be to implemented on a small microcontroller and also allowing the applications to be stand-alone systems.

The remainder of this paper is organized as follows: Section II describes the hardware and the mathematical model of the ball-and-plate system. Section III summarizes the method of solving QP problem by solving the NNLS problem. Section IV describes the implementation method and algorithm. Section V details the analysis of the experimental results and a comparison of tracking performance of MPC with that of other controllers. Finally, Section VI lists our conclusion.

II. BALL-AND-PLATE CONTROL SYSTEM

A. HARDWARE AND KINEMATICS

In this study, we use the laboratory-built ball-and-plate system that was used in [1]. Most of the existing ball-and-plate systems use a vision camera as a sensor [2], [3], [7]. To use a vision camera, the system is usually equipped with a computer to handle image processing, and the vision camera only captures fewer than 60 frames/s. In contrast, our laboratory-built system uses a touch panel to detect the position of the ball. The touch panel obtains the position data every millisecond and it does not require a high-performance computer.

Unlike other ball-and-plate systems, our laboratory-built system uses a Stewart platform for actuation so that the system has six degrees of freedom. This will be helpful for extending its usage in the future to generate more complex movements. Because we use six servo motors for the Stewart platform, the angles of each servo motor, δ_1 to δ_6, should be calculated by using inverse kinematics to rotate the plate to angles α and β. Then, the Stewart platform rotates the plate. As the plate rotates, the position of the ball also changes by x and y. This rotating procedure is shown in Fig. 2, and we use the kinematics analysis that is explained in [1]. In Fig. 2, α^d and β^d are the desired angles of the plate, α^r and β^r are the actual angles, and x and y are the position of the ball. By using this procedure, we can use the angle of the plate as control input and the position of the ball as output.

B. MATHEMATICAL MODEL

Fig. 3 shows the conceptual diagram for a ball-and-plate system. The linearized model of the system can be expressed as follows [1]:

$$\ddot{x} + \frac{5}{7} g\alpha = 0, \tag{1}$$

$$\ddot{y} + \frac{5}{7} g\beta = 0, \tag{2}$$

where g is the acceleration of gravity, α and β are the angle of the plate, and $\ddot{x}$ and $\ddot{y}$ are the acceleration of the ball with respect to the X and Y axes, respectively.

Equations (1) and (2) are decoupled, so we can consider them as two different single-input and single-output systems. Therefore, we can design a controller by considering only one axis and then apply it to both systems. The continuous state-space model of the ball-and-plate system is given

as follows [1]:

$$\begin{aligned} \dot{x} &= A_c x + B_c u, \\ y &= C_c x, \end{aligned} \tag{3}$$

where

$$\begin{aligned} A_c &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \ B_c = \begin{bmatrix} 0 \\ -\frac{5}{7}g \end{bmatrix}, \\ C_c &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \ x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \end{aligned} \tag{4}$$

Here, x_1 and x_2 are the position and velocity of the ball, respectively. Theoretically, this model can be used, but an offset problem can occur in the experiment. This is because the system recognizes the angle of the plate as zero when aligning the plate horizon with the base part of the system. Hence, we use an augmented model that includes an integration term as a new state variable. The augmented state-space model becomes

$$\begin{aligned} \dot{x} &= A_a x + B_a u, \\ y &= C_a x, \end{aligned} \tag{5}$$

where

$$\begin{aligned} A_a &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \ B_a = \begin{bmatrix} 0 \\ -\frac{5}{7}g \\ 0 \end{bmatrix}, \\ C_a &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \ x = \begin{bmatrix} x_1 \\ x_2 \\ \int x_1 \end{bmatrix}. \end{aligned} \tag{6}$$

III. NNLS-BASED MPC

In this section, we explain the method for solving the optimization problem of the MPC using the solution to the NNLS problem. Before solving the optimization problem, we convert the given model to a discrete state-space model as follows:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \\ y(k) &= Cx(k) + Du(k), \end{aligned} \tag{7}$$

where k is the current step, x is the state vector, u is the input vector, and y is the output vector. The matrices A, B, C, and D can be found easily by using the MATLAB function $c2d$. Generally, the given discrete model can be used for MPC. However, we expand the model and use the expanded model because the ball-and-plate system has servo motors and they have limitations in the input increment. The expanded model can be described as follows:

$$\begin{aligned} \bar{x}(k+1) &= \bar{A}\bar{x}(k) + \bar{B}\triangle u(k), \\ \bar{y}(k) &= \bar{C}x(k), \end{aligned} \tag{8}$$

where

$$\begin{aligned} \bar{A} &= \begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix}, \ \bar{B} = \begin{bmatrix} B \\ 0 \end{bmatrix}, \\ \bar{C} &= \begin{bmatrix} C & D \end{bmatrix}, \ \bar{x}(k) = \begin{bmatrix} x(k) \\ u(k) \end{bmatrix}. \end{aligned} \tag{9}$$

Here, $\triangle u(k)$ is the input increment such that $u(k) = u(k-1) + \triangle u(k)$. By using this model, the MPC can consider the feedforward term D in the prediction and also the model makes it easy to handle the limitation of an input increment.

A. OPTIMIZATION FOR THE MPC

The MPC solves the optimization problem at every sampling time according to the given cost function and the constraints. Here, the cost function can be expressed as

$$J = \sum_{i=1}^{m} \sum_{j=1}^{N_p} \sigma_i [r_i(k+j) - y_i(k+j)]^2 + \lambda \sum_{j=0}^{N_u - 1} [\triangle u(k+j)]^2, \tag{10}$$

where m is the dimension of the output, N_u is the control horizon, N_p is the prediction horizon, r is the reference, y is the output, σ is the weight on the tracking error, and λ is the weight on the input increment. Equation (10) can be expressed in vector form as follows:

$$J = (r - \hat{y})^T \Sigma (r - \hat{y}) + \lambda \triangle u^T \triangle u, \tag{11}$$

where

$$\begin{aligned} \Lambda &= \mathrm{diag}(\sigma_1, \ldots, \sigma_m), \\ \Sigma &= \mathrm{diag}(\Lambda, \Lambda, \ldots, \Lambda), \\ \triangle u &= \begin{bmatrix} \triangle u(k) & \ldots & \triangle u(k + N_u - 1) \end{bmatrix}^T, \\ \bar{r}_i &= \begin{bmatrix} r_1(k+i) \\ \vdots \\ r_m(k+i) \end{bmatrix}, \ \bar{y}_i = \begin{bmatrix} \hat{y}_1(k+i) \\ \vdots \\ \hat{y}_m(k+i) \end{bmatrix}, \\ r &= \begin{bmatrix} \bar{r}_1 \\ \bar{r}_2 \\ \vdots \\ \bar{r}_{N_p} \end{bmatrix}, \ \hat{y} = \begin{bmatrix} \bar{h}_1 \\ \bar{h}_2 \\ \vdots \\ \bar{h}_{N_p} \end{bmatrix}. \end{aligned} \tag{12}$$

Here $\hat{y}$ is the prediction of output and Σ is the weight matrix on the tracking error. The prediction of output $\hat{y}$ can be calculated from the following relation:

$$\begin{aligned} \hat{y} &= F\hat{x}(k) + H\triangle u(k), \\ F &= \begin{bmatrix} C\bar{A} \\ \vdots \\ C\bar{A}^{N_p} \end{bmatrix}, \ H = \begin{bmatrix} h_{1,1} & \ldots & h_{1,N_u} \\ \vdots & \ddots & \vdots \\ h_{N_p,1} & \ldots & h_{N_p,N_u} \end{bmatrix}, \\ h_{j,i} &= \begin{cases} C\bar{A}^{j-i}B, & j \geq i, \\ 0, & j < i. \end{cases} \end{aligned} \tag{13}$$

The first term of the predicted output is a free response, which means it is the predictive output when $\triangle u = 0$, and it can be simply expressed as

$$f = F\hat{x}(k). \tag{14}$$

By substituting (13) and (14) into (11), we can obtain the following objective function:

$$\begin{aligned} J(\triangle u) = \triangle u^T [H^T \Sigma H + \lambda I] \triangle u - 2(r - f)^T \Sigma^T H \triangle u \\ + (r - f)^T \Sigma (r - f). \end{aligned} \tag{15}$$

The term $(r-f)^T\Sigma(r-f)$ is a constant, which does not affect the solution to the optimization problem. Therefore, the solution to the optimization with constraint can be expressed as follows:

$$\begin{aligned}\triangle u^* &= \min_{\triangle u}\left\{\triangle u^T[H^T\Sigma H+\lambda I]\triangle u - 2(r-f)^T\Sigma^T H\triangle u\right\}\\ &= \min_{\triangle u}\left\{\frac{1}{2}\triangle u^T[H^T\Sigma H+\lambda I]\triangle u - (r-f)^T\Sigma^T H\triangle u\right\},\\ &\quad \text{subject to } \underline{u}\le u\le \bar{u},\ \triangle\underline{u}\le\triangle u\le\triangle\bar{u},\end{aligned} \tag{16}$$

where $\bar{u}$ and $\underline{u}$ are the maximum and minimum of the input, respectively, and $\Delta\bar{u}$ and $\Delta\underline{u}$ are that of the input increment, respectively. Consequently, the optimal control increment $\triangle u^*$ can be calculated by solving the above optimization problem.

B. QP ALGORITHM USING THE NNLS SOLUTION

The general form of the QP problem is given as follows:

$$x^* = \min_x\left\{\frac{1}{2}x^TPx + q^Tx\right\},\ \text{subject to } A_\eta x\le b_\eta, \tag{17}$$

where P is the symmetric matrix, q is a vector, and A_η and b_η are the constraint matrix and vector, respectively. Suppose P is positive definite. Then the QP problem becomes strictly a convex optimization, which guarantees the existence of a global unique solution. We define a cost function for the least squares with the inequality (LSI) problem as

$$\hat{J}(x) = ||\sqrt{P}x + \sqrt{P^{-1}}q|| > 0. \tag{18}$$

According to

$$J(x) = \frac{1}{2}[\hat{J}^2(x) - q^TP^{-1}q], \tag{19}$$

J is a monotonically increasing function with respect to the function $\hat{J}$, so the argument that minimizes $\hat{J}$ also minimizes J.

Therefore, the QP problem can be converted to the LSI problem as follows:

$$\begin{aligned}&\min \hat{J}(x) = ||\sqrt{P}x + \sqrt{P^{-1}}q||,\\ &\text{subject to } A_\eta x\ge b_\eta.\end{aligned} \tag{20}$$

According to [14], [15], the LSI problem can be reformulated to the LDP problem. Additionally, it has been proven that the LDP can be converted to the equivalent NNLS problem. In this study, we apply this method to solve the QP problem.

Suppose $\sqrt{P} = USV^T$ by singular value decomposition. Because P is positive definite, there always exists the inverse of S. Besides, U is an orthogonal matrix so that we can express $\hat{J}(x)$ as follows:

$$\hat{J}(x) = ||USV^T(x+P^{-1}q)|| = ||SV^T(x+P^{-1}q)||. \tag{21}$$

Let

$$\begin{aligned}z &= SV^T(x+P^{-1}q),\\ A_z &= -A_\eta VS^{-1},\\ b_z &= -A_\eta P^{-1}q - b_\eta.\end{aligned} \tag{22}$$

Then, by using (22), we can convert the LSI problem into the equivalent LDP problem as follows [15]:

$$\begin{aligned}&\min_z ||z||,\\ &\text{subject to } A_zz\ge b_z.\end{aligned} \tag{23}$$

Suppose z^* is the solution to (23); then, we can obtain z^* from the solution to the following NNLS problem:

$$\begin{aligned}&\min_d ||Qd-\gamma||,\\ &\text{subject to } d\ge 0,\\ &Q = \begin{bmatrix}A_z^T\\ b_z^T\end{bmatrix},\ \gamma = \begin{bmatrix}0\\ \vdots\\ 0\\ 1\end{bmatrix}.\end{aligned} \tag{24}$$

Suppose d^* is the solution to (24). Then the solution to the LDP is given by the following equation [15]:

$$z^* = \frac{A_z^Td^*}{1-b_z^Td^*}. \tag{25}$$

Consequently, the solution to the original QP problem can be derived from (22) and (25) as follows:

$$x^* = VS^{-1}z^* - P^{-1}q. \tag{26}$$

C. CONSTRAINTS

According to (16) and (17), we set $P = H^T\Sigma H + \lambda I$, $q = H\Sigma(r-f)$, and $x = \triangle u$. The constraint matrices A_η and b_η consider the constraints on the input and input increments. Define

$$\begin{aligned}I_D &= \begin{bmatrix}I_m & 0 & \dots & 0\\ 0 & I_m & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \dots & I_m\end{bmatrix},\\ I_L &= \begin{bmatrix}I_m & 0 & \dots & 0\\ I_m & I_m & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ I_m & I_m & \dots & I_m\end{bmatrix},\ I_V = \begin{bmatrix}I_m\\ I_m\\ \vdots\\ I_m\end{bmatrix},\end{aligned} \tag{27}$$

where I_m is the $m\times m$ identity matrix, and I_D, I_L, and I_V have proper dimensions with respect to the horizon length. If we suppose ΔU is the input sequence, which is going to be the solution to the QP problem, then the constraints can be expressed as follows:

$$\begin{bmatrix}I_D\\ -I_D\\ I_L\\ -I_L\end{bmatrix}\Delta U \le \begin{bmatrix}I_V\bar{u}\\ -I_V\underline{u}\\ I_V(\bar{u}-u_{k-1})\\ -I_V(\underline{u}-u_{k-1})\end{bmatrix}. \tag{28}$$

From the above equation, we can obtain

$$A_\eta = \begin{bmatrix}I_D\\ -I_D\\ I_L\\ -I_L\end{bmatrix},\ b_\eta = \begin{bmatrix}I_V\bar{u}\\ -I_V\underline{u}\\ I_V(\bar{u}-u_{k-1})\\ -I_V(\underline{u}-u_{k-1})\end{bmatrix}. \tag{29}$$

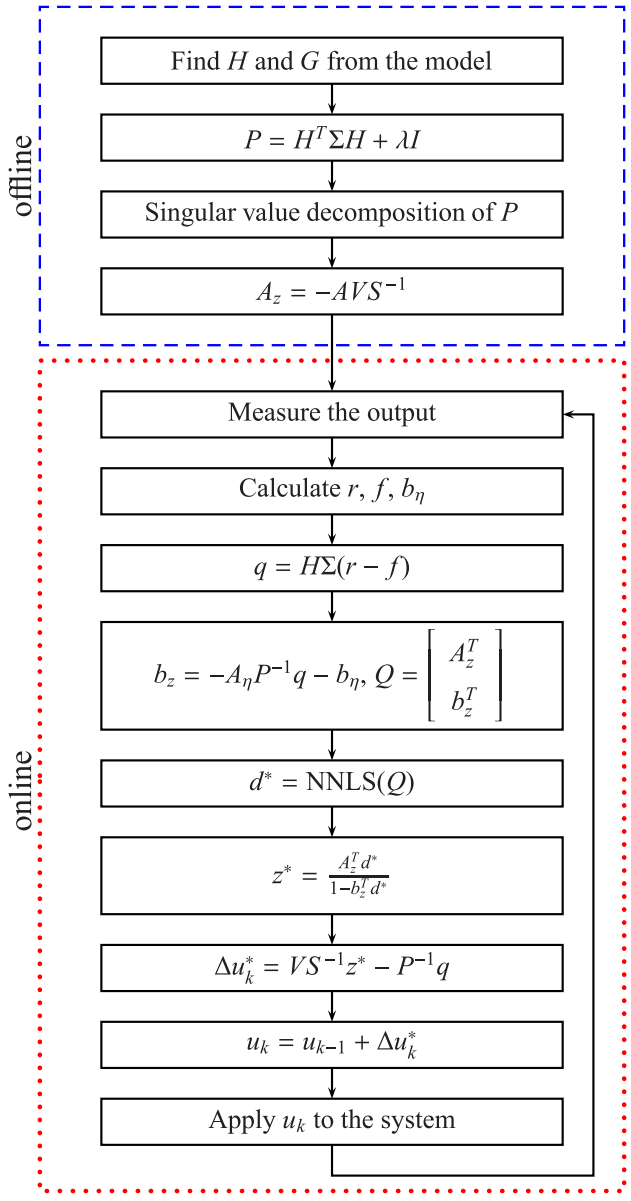


FIGURE 4. Block diagram of the optimizer block.

IV. IMPLEMENTATION

A. ALGORITHM

This section includes our main contribution of the implementation algorithm. The proposed method for solving the QP problem requires complex tasks such as solving the NNLS problem, singular value decomposition, and the inversion of matrices. However, because we use a time-invariant system, the matrices P and A_η are fixed and only q and b_η change at every sampling time. Therefore, the complexity of the calculations can be reduced by using these characteristics.

We isolate the offline tasks from the online tasks. Because P is fixed, singular value decomposition of P and matrix inversion can be calculated offline. Then, the online calculation includes the basic multiplication of matrices and vectors and calculation of the NNLS problem. Fig. 4 shows these process in detail.

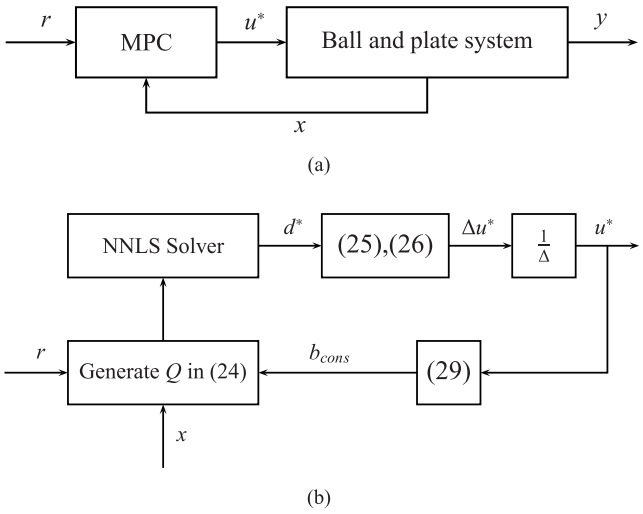


FIGURE 5. Block diagram of the control process. (a) Diagram of the control system. (b) Block diagram of the proposed MPC.

We build a code generator using MATLAB. The code generator conducts the offline calculation that is shown in Fig. 4. Then, it generates a header file that includes all the data. By using the generated header file, we can simply include it and use the data in a microcontroller.

For an online process, we implement only an NNLS solver and matrix multiplication in a C program. The NNLS solver is widely used with C-code applications and it is available on the website [13].

Figs. 5(a) and 5(b) show the block diagrams of the model predictive control strategy for the ball-and-plate system. Fig. 5(a) show the whole control process for a ball-and-plate system and Fig. 5(b) shows the MPC process in detail. By using this procedure, we can calculate the optimal input sequence u^* and then apply the first optimal input of the sequence to the system. This procedure is conducted repeatedly.

B. MICROCONTROLLER

We use a Nucleo-32 board, which is small and a low-cost open-source piece of hardware. The Nucleo-32 board has a maximum clock speed of 72 MHz. Compared to the Intel Celeron processor (266 MHz) that is used in [16], the Nucleo-32 board is limited not only in terms of clock speed but also in terms of memory size. Nevertheless, the proposed method can be applied to the Nucleo-32 board because the algorithm takes this limitation into account.

V. EXPERIMENT

In this section, we demonstrate the performance of the MPC on a ball-and-plate system through simulations and experiments.

A. SIMULATION

Compared to the other control theories that are applied to the ball-and-plate system such as LQ tracking and SMC [1], the biggest advantage of using MPC is that it considers

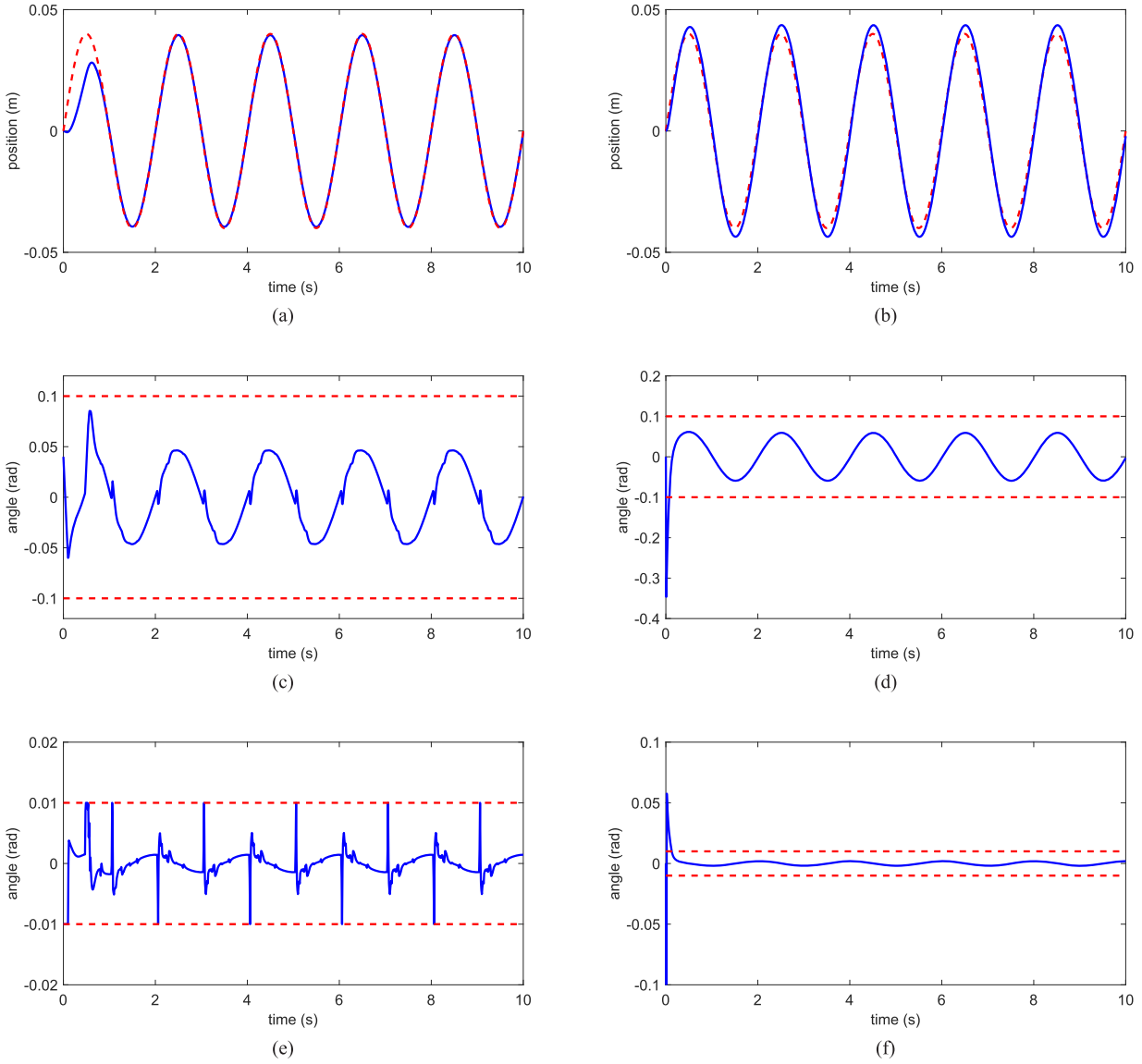


FIGURE 6. Sinusoidal reference tracking with a period of 2 s. (a) Position of the ball using MPC. (b) Position of the ball using SMC. (c) Control input of the MPC. (d) Control input of the SMC. (e) Input increment of the MPC. (f) Input increment of the SMC.

the future reference and tracks it while meeting the constraints. To compare performances, we conducted simulations of SMC and MPC to track a sinusoidal reference. We set the same parameters for SMC as those used in [1]. The parameters for the MPC were set as follows: $m = 3$, $N_p = 10$, $N_u = 5$, $\begin{bmatrix} \sigma_1 & \sigma_2 & \sigma_3 \end{bmatrix} = \begin{bmatrix} 1.0 & 1.2 & 0.8 \end{bmatrix}$, and $\lambda = 0.5$. We assumed that the ball-and-plate system has a limit on the input of 0.1 and a limit on the input increment of 0.01. The results are shown in Fig. 6. Figs. 6(a), 6(c), and 6(e) show the position of the ball, control input, and input increment, respectively, for the MPC case. Figs. 6(b), 6(d), and 6(f) show the position of the ball, control input, and input increment, respectively, for the SMC case. As shown in Figs. 6(a) and 6(b), the maximum error of SMC is larger than that of MPC, which means that the reference varies too fast for SMC to track it. Figs. 6(c) and 6(e) show that MPC generates the control input that meets the constraints as well as the increment constraint. However, SMC generates the input that goes beyond the limit as shown in Figs. 6(d) and 6(f).

B. EXPERIMENT

To illustrate the tracking performance of MPC, we conducted three experiments under the same condition with different controllers. We used SMC and LQ tracking control and set the same design parameters as those used in [1]. Before the experiment, we set the same parameters as used in the simulation to execute the offline process for MPC. As previously mentioned in the simulation, we set the prediction horizon to 10 and the control horizon to 5. These horizon lengths are set to use the maximum amount of memory allowed by the Nucleo-32 board.

To ensure real-time performance of MPC on the embedded system, we measured the time taken to execute the control loop once. We used the GPIO function of Nucleo-32 and

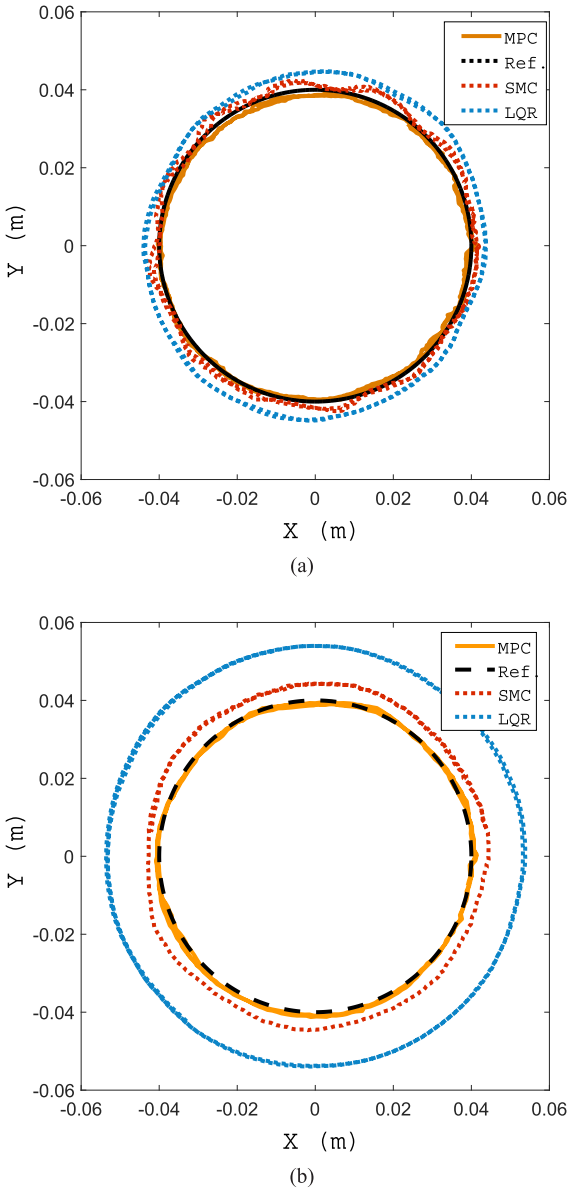


FIGURE 7. Trajectories generated by three different controllers on the *X*–*Y* plane. (a) Circle with a period of 8 s. (b) Circle with a period of 4 s.

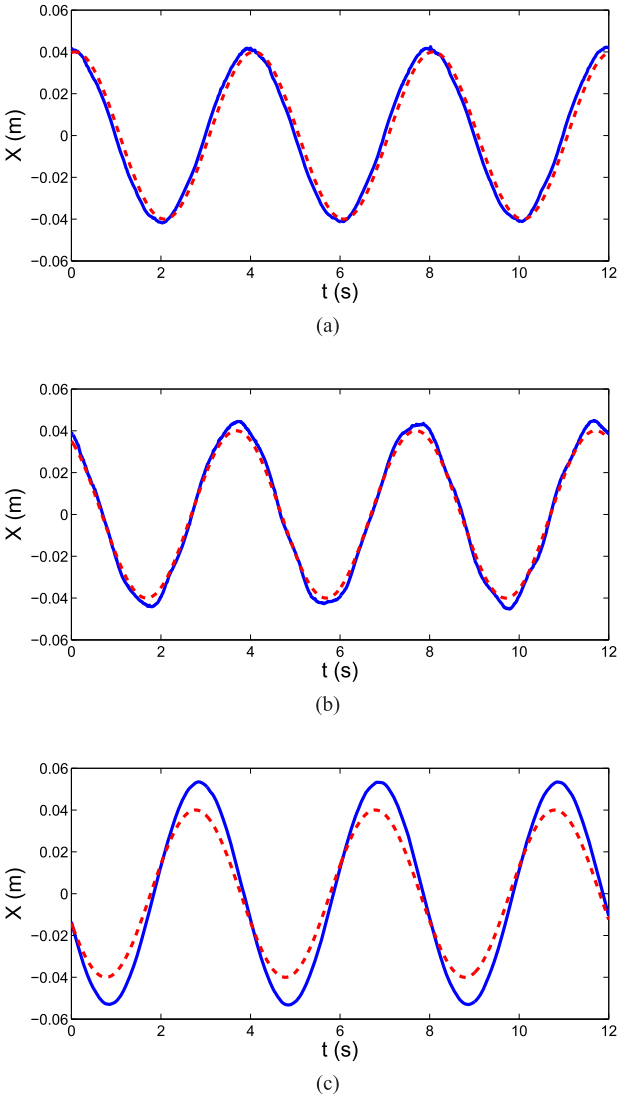


FIGURE 8. Comparison of tracking performance with the period of 4 s on the *x* axis. (a) MPC result. (b) SMC result. (c) LQ tracking result.

generated a pulse that measured the time taken from sensing the position to applying the calculated control input. The control time was measured from the minimum of 3.3 ms to the maximum of 6.1 ms. Therefore, it is possible to use a sampling time of 10 ms. However, we used a sampling time of 20 ms, because the Nucleo-32 board cannot predict enough time owing to memory limitations. At this time, the system can predict ∼0.2 s.

Fig. 7 depicts the tracking performance of a circular trajectory on the *X*–*Y* plane and Fig. 8 shows the tracking performance on the *x* axis. These figures are taken at a steady-state condition. The amplitude of the sinusoidal reference, which is the radius of the circle trajectory in Fig. 7, is 4 cm and the period is 4 s. As shown in Figs. 7(a) and 7(b), every controller shows poor performances as the period becomes shorter. Similarly, even though we use the same period, poor performances occur if the radius of the circle becomes larger. However, the performance of MPC is still acceptable because it considers the dynamic movement of the future. In Fig. 8, the maximum errors MPC, SMC, and LQ tracking control are approximately 1, 5, and 13 mm, respectively.

Most of the ball-and-plate systems [1]–[7] exhibited good performances, but only with the limitation of the amplitude and period in the reference trajectory. They do not guarantee good performances with a larger and faster reference trajectory. In this sense, the proposed MPC on the ball-and-plate system is effective because the proposed controller can cover much bigger and faster references than other controllers.

VI. CONCLUSIONS

In this paper, we proposed an MPC implementation method that runs on a microcontroller and applied it to a laboratory-built ball-and-plate system to improve tracking performance. We enhanced the calculation efficiency by separating

the offline calculation from the online calculation so that MPC can run on a Nucleo-32 microcontroller in real time. We used the C-code application of the NNLS solver to construct the QP solver that can be easily adapted to the microcontroller. Through the experiment, we verified the tracking performance of MPC compared to that of other controllers that have been widely used for ball-and-plate systems.

The proposed method enables MPC implementation in a limited performance microcontroller. Furthermore, it is advantageous because tracking performance is greatly improved by actually implementing MPC, which has not been covered in most studies of ball-and-plate system control.

REFERENCES

[1] H. Bang and Y. S. Lee, "Implementation of a ball and plate control system using sliding mode control," *IEEE ACCESS*, vol. 6, no. 1, pp. 32401–32408, 2018.

[2] C. J. Bay and B. P. Rasmussen, "Exploring controls education: A re-configurable ball and plate platform kit," in *Proc. Amer. Control Conf. (ACC)*, Boston, MA, USA, Jul. 2016, pp. 6652–6657.

[3] L. Spacek, V. Bovál, and J. Vojtesek, "Digital control of Ball & Plate model using LQ controller," in *Proc. 21st Int. Conf. Process Control (PC)*, Strbske Pleso, Slovakia, Jun. 2017, pp. 36–41.

[4] H. Liu and Y. Liang, "Trajectory tracking sliding mode control of Ball and Plate system," in *Proc. 2nd Int. Asia Conf. Inform. Control, Automat. Robot. (CAR)*, Wuhan, China, vol. 3, Mar. 2010, pp. 142–145.

[5] X. Fan, N. Zhang, and S. Teng, "Trajectory planning and tracking of ball and plate system using hierarchical fuzzy control scheme," *Fuzzy Sets Syst.*, vol. 144, no. 2, pp. 297–312, 2004.

[6] R. K. Pour, H. Khajvand, and S. A. A. Moosavian, "Fuzzy logic trajectory tracking control of a 3-RRS ball and plate parallel manipulator," in *Proc. 4th Int. Conf. Robot. Mechatron. (ICROM)*, Tehran, Iran, Oct. 2016, pp. 343–348.

[7] Z. Fei, Q. Xiaolong, L. Xiaoli, and W. Shangjun, "Modeling and PID neural network research for the ball and plate system," in *Proc. Int. Conf. Electron., Commun. Control (ICECC)*, Ningbo, China, Sep. 2011, pp. 331–334.

[8] M. Oravec and A. Jadlovská, "Model predictive control of a ball and plate laboratory model," in *Proc. IEEE 13th Int. Symp. Appl. Mach. Intell. Inform. (SAMI)*, Herl'any, Slovakia, Jan. 2015, pp. 165–170.

[9] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Mar. 2010.

[10] Z. Li, J. Deng, R. Lu, Y. Xu, H. Bai, and C.-Y. Su, "Trajectory-tracking control of mobile robot systems incorporating neural-dynamic optimized model predictive approach," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 46, no. 6, pp. 740–749, Jun. 2015.

[11] J. C. Lins Barreto S, A. G. S. Conceicao, C. E. T. Dorea, L. Martinez, and E. R. De Pieri, "Design and implementation of model-predictive control with friction compensation on an omnidirectional mobile robot," *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 2, pp. 467–476, Apr. 2014.

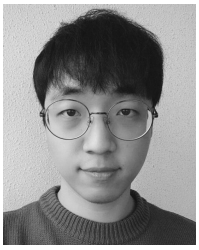
[12] J. Koenemann *et al.*, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Hamburg, Germany, Sep./Oct. 2015, pp. 3346–3351.

[13] *NNLS File on a Web-Site*. Accessed: Aug. 3, 2018. [Online]. Available: https://hesperia.gsfc.nasa.gov/~schmahl/nnls/nnls.c

[14] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, vol. 15. Philadelphia, PA, USA: SIAM, 1995.

[15] Y.-S. Lee, G.-Y. Gyeong, and J.-H. Park, "QP solution for the implementation of the predictive control on microcontroller systems and its application method," (in Korean), *J. Inst. Control, Robot. Syst.*, vol. 20, no. 9, pp. 908–913, 2014.

[16] S. Richter, "Computational complexity certification of gradient methods for real-time model predictive control," Ph.D. dissertation, Dept. Inf. Technol. Elect. Eng., ETH Zürich, Zürich, Switzerland, 2012.



HEESEUNG BANG received the B.S. degree in mechanical engineering from Inha University, Incheon, South Korea, in 2018, where he is currently pursuing the M.S. degree in electrical engineering. His research interests include nonlinear systems, predictive control, embedded systems, and robotics.

YOUNG SAM LEE received the B.S. and M.S. degrees from Inha University, Incheon, South Korea, in 1999, and the Ph.D. degree from Seoul National University, South Korea, in 2003, all in electrical engineering. From 2003 to 2004, he was a Senior Researcher with Samsung Electronics Co. Since 2004, he has been with the Department of Electrical Engineering, Inha University. He has authored four books and more than 50 articles. His research interests include computer-aided control system designs, rapid control prototyping, control and instrumentation, robot engineering, and embedded systems.