Journal of Institute of Control, Robotics and Systems (2014) 20(9):908-913
http://dx.doi.org/10.5302/J.ICROS.2014.14.0049 ISSN:1976-5622 eISSN:2233-4335

예측제어의 마이크로콘트롤러 구현을 위한 QP 해법과 그 적용방법

QP Solution for the Implementation of the Predictive Control on Microcontroller Systems and Its Application Method

이 영 삼*, 경 기 영, 박 재 헌
(Young-Sam Lee[1,*], Gi-Young Gyeong[1], and Jae-Heon Park[1])
[1]Department of Electrical Engineering, Inha University

Abstract: In this paper, we propose a method by which QP (Quadratic Programming) problems can be solved in realtime so that we can implement the predictive control algorithm on a microcontroller system. Firstly, we derive a solution to QP problems by converting the original QP problems to its equivalent least squares with inequalities. Secondly, we propose a predictive control algorithm that can give good realtime computation performance by utilizing the fact that some parameters needed for solving QP problems can be computed offline. Finally, we illustrate that the proposed method can give good realtime features by running the C-code application constructed using the proposed method on a microncontroller system.

Keywords: quadratic programming, predictive control, microcontroller system, realtime

I. 서론

예측 제어는 산업계 및 학계에서 많은 관심을 받아왔고 특히 응답이 비교적 느린 공정 시스템의 제어에 적합하다고 알려져 있다. 예측 제어는 모델 정보를 이용하여 시스템의 미래 출력을 예측하고 예측된 출력에 의해 발생한 비용함수가 최소가 되게끔 시스템의 제어입력을 최적화 연산을 통해 계산하는 제어방식이다. 예측제어는 가정하고 있는 모델식의 종류에 따라 다양한 형태로 연구되었다. 그중 대표적인 기법들로는 ARMA 모델을 가정하는 GPC [3,4], 스텝 응답 모델을 사용하는 DMC (Dynamic Matrix Control) [5], 상태공간 모델을 사용하는 MPC (Model Predictive Control) [10] 등이 있다. 예측제어는 비용함수를 최소로 만든다는 최적제어의 개념과 더불어 입력과 출력에 나타나는 제약조건을 조직적으로 고려할 수 있는 장점을 가진다는 특징이 있다. 예측제어의 해를 구하는 문제는 선형행렬부등식(LMI: Linear Matrix Inequality) 문제로 귀결되는 일부 예측제어기법[7]을 제외하고는 대부분 QP 문제로 귀결된다. 따라서 QP 문제의 해법은 예측제어기의 구현에 필수적인 요소이다.

많은 논문들이 예측제어의 이론 또는 응용문제를 다루었다. 그리고 모의실험을 통해 효용성을 검증하는 접근법을 쓰고 있다. 하지만 예측제어를 C-code 등으로 구현하여 실제 마이크로콘트롤러에 탑재하는 것과 관련된 연구는 상대적으로 관심을 많이 받지 못했다. 한가지 이유로는 예측제어 알고리즘을 구현하기 위해 필요한 QP의 해법에 대한 C-code 알고리즘을 마이크로콘트롤러에 탑재할 만큼 간결한 것을 구하기 힘들고 QP 문제의 풀이에 소요되는 시간이 길어서 비교적 빠른 시스템에는 실시간 적용가능성이 떨어질 것으로 생각하기 때문일 것이다.

최근에는 실시간으로 QP를 푸는 대신 offline 계산을 통해서 예측제어의 해를 모든 상태변수에 대하여 영역별로 미리 구해 놓고 실시간 연산에서는 상태변수가 속한 영역을 검사한 후 그 영역에서의 제어법칙을 적용하는 방식의 연구들이 제안되었다[2,9,13]. 하지만 이러한 방식은 상태공간모델을 사용하는 예측제어에만 적용할 수 있다는 단점이 있다.

이 논문에서는 Matlab 또는 CEMTool과 같은 소프트웨어를 사용하여 모의실험으로만 검증해 보았던 예측제어 알고리즘을 실제로 마이크로콘트롤러에 탑재하여 시스템을 실시간 제어하는데 사용할 수 있게끔 하는 토대를 제공하는 것을 목표로 한다. 이를 위해 이 논문에서는 특이값 분해를 이용하여 QP 문제의 해를 제안하고 제안된 QP 문제의 해를 예측제어를 실시간 적용하는데 활용할 수 있게끔 계산속도를 향상하는 방법을 제안한다.

논문의 구성은 다음과 같다. II 절에서는 특이값 분해를 이용한 QP 문제의 해를 제시한다. III 절에서는 예측제어 문제가 QP 문제로 설정되는 것을 기술한다. IV 절에서는 제안된 QP의 해를 예측제어 알고리즘에 적용하는데 있어 계산속도를 향상하는 방법을 제안한다. V 절에서는 제안된 방법을 적용한 예측제어 알고리즘을 마이크로콘트롤러 상에서 구동함으로써 제안된 방법을 통해 예측제어 알고리즘에 대한 실시간 계산성능을 확보할 수 있음을 예시한다. 마지막으로 VI 절에서 결론을 맺는다.

* Corresponding Author
Manuscript received May 12, 2014 / revised June 17, 2014 / accepted June 19, 2014
이영삼, 경기영, 박재헌: 인하대학교 전기공학과
(lys@inha.ac.kr/jihad12@korea.com/mihonyang@gmail.com)
※ 본 연구는 미래창조과학부 및 정보통신산업진흥원의 IT융합고급인력과정지원사업의 연구결과로 수행되었으며(NIPA-2014-H0401-14-1003) 한국전력공사의 재원으로 기초전력연구원의 2013년 선정 기초연구개발과제의 지원을 받아 수행된 것임(과제번호: R13GA04).

II. 특이값 분해를 이용한 QP 문제의 해

QP(Quadratic programming) 문제는 다음과 같이 기술된다.

$$Minimize\, J(x) = \frac{1}{2}x^T Px + q^T x \text{ subject to } Ax \le b \quad (1)$$

$P \ge 0$이면 QP 문제는 convex가 되어 전역해(global solution)가 보장되고 $P > 0$이면 유일해가 존재한다. 예측제어에서의 제어입력 계산 문제는 $P > 0$인 QP 문제로 귀결된다. 이 논문에서는 $P > 0$인 경우에 대한 QP 문제의 해를 제시하고 이를 예측제어 알고리즘에 적용할 때 계산효율성을 향상시키는 방법을 제안한다.

먼저 $P > 0$인 경우의 QP 문제에 대한 해를 유도하자. 다음과 같은 LSI (Least Squares with Inequality) 문제를 생각해 보자.

$$Minimize\, \hat{J}(x) = \| \sqrt{P}x + \sqrt{P^{-1}}q \| \quad (2)$$
$$\text{subject to } Ax \le b$$

여기서 비용함수 $\hat{J}(x)$에 대해

$$\hat{J}^2(x) = x^T Px + 2q^T x + q^T P^{-1} q$$

이 성립하므로 QP 문제의 비용함수 $J(x)$와 LSI 문제의 비용함수 $\hat{J}(x)$는 다음의 관계가 성립한다.

$$J(x) = \frac{1}{2}[\hat{J}^2(x) - q^T P^{-1} q]$$

$\hat{J}(x) \ge 0$이고 J는 $\hat{J}$에 대해 단조 증가함수이므로 결국 $\hat{J}$가 최소일 때 J도 최소값을 갖게 된다. 즉 (1)과 (2)에 주어진 문제는 등가의 문제가 된다. 즉 원래의 QP 문제가 (2)와 같은 LSI 문제로 등가변형됨을 알 수 있다. [9]에서 LSI 문제는 LDP (Least Distance Programming)로 등가변형후 NNLS (Nonnegative Least Squares) 문제로 최종적으로 변형할 수 있음이 증명되었다. 이 논문에서는 이를 이용하여 $P > 0$인 경우의 QP 문제에 대한 풀이법을 다음과 같이 제시한다.

$\bar{q} = P^{-1}q$라고 하면 $\hat{J} = \| \sqrt{P}(x + \bar{q}) \|$로 나타낼 수 있다. 특이값 분해를 이용하여 $\sqrt{P} = USV^T$로 나타낼 수 있다고 가정해보자.

$P > 0$이므로 S는 항상 역행렬이 존재한다. U는 단위직교행렬이므로 이로부터 다음이 성립함을 알 수 있다.

$$\hat{J}(x) = \| USV^T(x + \bar{q}) \| = \| SV^T(x + \bar{q}) \|$$

여기서 $z = SV^T(x + \bar{q})$, $\bar{A} = -AVS^{-1}$, $\bar{b} = -A\bar{q} - b$라고 하면 최적화 문제는 다음과 같은 LDP (Least Distance Programming) 문제로 변형된다.

$$Minimize \quad \| z \| \text{ subject to } \bar{A}z \ge \bar{b}$$

LDP 문제의 해를 z^*라고 했을 때 z^*는 다음과 같은 NNLS (Nonnegative Least-Squares) 문제를 풀어 그 해를 통해 구할 수 있다[9].

$$Minimize \quad \| Qd - r \| \text{ subject to } d \ge 0$$
$$Q = \begin{bmatrix} \bar{A}^T \\ \bar{b}^T \end{bmatrix}, \quad r = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

NNLS 문제의 해를 d^*라고 하면 LDP 문제의 해는 다음과 같이 구해진다.

$$z^* = \frac{\bar{A}^T d}{(1 - b^T d)}$$

z^*로부터 원래의 QP 문제의 해 x^*를 다음과 같이 구할 수 있다.

$$x^* = VS^{-1}z^* - \bar{q} = VS^{-1}z^* - P^{-1}q$$

P, q, A, b가 주어졌을 때 QP의 해를 얻기 위해 필요한 계산들을 열거해 보면 다음과 같다.

(a) 행렬간의 곱, 행렬과 벡터의 곱, 벡터의 덧셈 및 뺄셈
(b) $\sqrt{P}$에 대한 특이값 분해
(c) P^{-1}를 계산하기 위한 역행렬 계산
(d) NNLS의 풀이

이중에서 (a)에 대한 계산은 C-code로 쉽게 구현가능하다. 하지만 (b), (c), (d)에 대한 계산을 C-code로 구현하기 위해서는 비교적 긴 알고리즘이 필요하다. (b)와 (c)에 해당하는 계산은 둘다 행렬의 특이값 분해를 통해 계산할 수 있고 NNLS의 해는 [9]에서 제시된 알고리즘을 사용하여 계산할 수 있다.

(b)와 (c)의 계산이 특이값 분해를 통해 계산가능한 것을 살펴보자. P가 다음과 같이 특이값 분해된다고 가정하자.

$$P = U\Sigma V^T$$

$P > 0$이므로 $\Sigma > 0$이고 $U = V$가 성립한다. U는 단위직교행렬이므로 $UU^T = I$

$$U\sqrt{\Sigma}\, V^T(U\sqrt{\Sigma}\, V^T) = U\Sigma V^T = P$$

로부터 $\sqrt{P}$는 다음과 같이 나타낼 수 있다.

$$\sqrt{P} = USV^T,\ S = \sqrt{\Sigma}$$

Σ는 대각행렬이므로 S 역시 대각행렬이고 주대각선 성분은 Σ의 주대각선 성분에 대한 제곱근의 값을 갖기 때문에 쉽게 계산할 수 있다. P^{-1}를 계산하기 위해서는 LU 분해에 이은 back substitution 방법이나 Cramer rule과 같은 역행렬 계산 방법을 사용할 수 있지만 여기서는 P의 특이값 분해를 계속 사용하여 계산할 수 있다. 다음과 같이

$$U\Sigma V^T(V(\Sigma)^{-1}U^T) = I$$

이 성립하므로 $P^{-1} = V(\Sigma)^{-1}U^T$ 임을 알 수 있다. Σ^{-1} 역시 주대각선 성분에 대한 역수만을 계산하면 되므로 쉽게 계산이 가능하다.

III. 예측제어 문제의 QP 문제로의 설정

변수에 사용된 아래첨자는 시점을 나타내는 것으로 가정하고 현재 시점을 k라고 가정하자. 입력을 u라고 한다면 u_{k+j}는 현재 시점으로부터 j 스텝 이후의 입력을 의미한다. 예측제어에서는 매 스텝마다 다음과 같이 제약조건을 만족하면서 주어진 비용함수를 최소화하는 제어량을 계산한다.

$$Minimize\ J=\sum_{j=1}^{N_p}\| w_{k+j}-\hat{y}_{k+j}\|^2+\sum_{j=0}^{N_c-1}\| \triangle u_{k+j}\|_{\Lambda}^2 \quad (3)$$

$$\text{subject to } u_{\min}\le u_{k+j}\le u_{\max},\ \triangle u_{\min}\le \Delta u_{k+j}\le \triangle u_{\max} \quad (4)$$

여기서 w는 기준입력, $\hat{y}$는 예측 출력, $\triangle u$는 제어증분(control increment)를 나타내며 N_p는 예측구간(prediction horizon), N_c는 제어구간(control horizon)이다. 일반적으로 $N_p \ge N_c$ 이고 Λ는 제어증분에 대한 가중치를 나타내는 대각행렬이다. 예측제어에서는 매 스텝마다 $k+N_c-1$까지 제어증분량을 계산하지만 사용되는 것은 $\triangle u_k$로써 실제 시스템에 사용되는 제어입력과는

$$u_k = u_{k-1}+\triangle u_k$$

의 관계가 성립한다. 현재로부터 N_p 스텝 이후까지의 예측출력을 행렬과 벡터를 이용하여 나타내면 다음과 같이 나타낼 수 있다.

$$\hat{Y}_k = G\triangle U_k + f_k \quad (5)$$

여기서

$$\hat{Y}_k = \left[\hat{y}_{k+1}^T\ \hat{y}_{k+2}^T\ \cdots\ \hat{y}_{k+N_p}^T\right]^T$$

$$\triangle U_k = \left[\triangle U_k^T\ \triangle U_{k+1}^T\ \cdots\ \triangle U_{k+N_c-1}^T\right]$$

이고 G는 시스템의 모델정보를 이용하여 계산한 예측행렬이고, f_k는 free response로써 과거의 data가 미래의 출력에 기여하는 양을 나타낸다. 여기서 G에는 아래 첨자가 붙지 않았는데 이것은 매 스텝마다 G의 값이 변하지 않기 때문이다. G와 f_k는 DMC, GPC, MPC 등 예측제어의 종류에 따라 각각의 계산 방식이 존재한다. 이 논문의 후반부에서는 수치예제를 위하여 DMC를 사용할 것이다. 미래의 기준입력 벡터를

$$W_k = \left[w_{k+1}^T\ w_{k+2}^T\ \cdots\ w_{k+N_p}^T\right]^T$$

로 정의하면 예측제어에서 최소화하고자 하는 비용함수는 $\triangle U_k$의 함수가 되어 다음과 같이 나타낼 수 있다.

$$J(\triangle U_k) = (W_k-\hat{Y}_k)^T(W_k-\hat{Y}_k)+\triangle U_k^T\Xi\triangle U_k$$

여기서

$$\Xi = diag[\Lambda,\ \Lambda,\ \cdots,\ \Lambda]$$

비용함수를 풀어서 $\triangle U_k$에 대한 2차형으로 정리하면 다음과 같다.

$$J(\triangle U_k) = \triangle U_k^T[G^TG+\Xi]\triangle U_k - 2(W_k-f_k)^TG\triangle U_k + (W_k-f_K)^T(W_k-f_k) \quad (6)$$

시스템이 갖는 입력 제약조건을 다시 나타내보자. 먼저 다음과 같이 I_D, I_L, I_V를 정의하자.

$$I_D=\begin{bmatrix} I_m & 0 & \cdots & 0\\ 0 & I_m & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & I_m\end{bmatrix},\ I_L=\begin{bmatrix} I_m & 0 & \cdots & 0\\ I_m & I_m & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ I_m & I_m & \cdots & I_m\end{bmatrix},\ I_V=\begin{bmatrix} I_m\\ I_m\\ \vdots\\ I_m\end{bmatrix}$$

여기서 m은 입력의 차원이고, I_D, I_L, I_V는 예측구간 및 제어구간의 길이에 따라 적절한 차원을 가진다고 가정하자. (4)에 주어진 입력에 대한 제약조건을 행렬과 벡터를 이용하여 나타내면

$$I_D\triangle U_k \le I_V\triangle U_{\max},\ \ -I_D\triangle U_k \le -I_V\triangle U_{\min}$$

$$I_L\triangle U_k \le I_V(u_{\max}-u_{k-1}),\ \ -I_L\triangle U_k \le -I_V(u_{\min}-u_{k-1})$$

이 되며 이것을 $Ax\le b$의 형태로 나타내면

$$\begin{bmatrix} I_D\\ -I_D\\ I_L\\ -I_L\end{bmatrix}\triangle U_k \le \begin{bmatrix} I_V u_{\max}\\ -I_V u_{\min}\\ I_V(u_{\max}-u_{k-1})\\ -I_V(u_{\min}-u_{k-1})\end{bmatrix}$$

로 나타낼 수 있다. 비용함수 (6)에서 마지막 항인 $(W_k-f_k)^T(W_k-f_k)$는 $\triangle U_k$의 함수가 아니므로 최적화와는 무관하다. 따라서 제약조건 (4)를 만족시키며 비용함수 (6)을 최소화시키는 해 $\triangle U_k^*$는

$$P = G^TG+\Xi,\ q=-2(W_k-f_k)^TG,$$

$$A=\begin{bmatrix} I_D\\ -I_D\\ I_L\\ -I_L\end{bmatrix},\ b=\begin{bmatrix} I_V u_{\max}\\ -I_V u_{\min}\\ I_V(u_{\max}-u_{k-1})\\ -I_V(u_{\min}-u_{k-1})\end{bmatrix}$$

인 QP 문제의 해가 된다. (7)에서 알 수 있듯이 예측제어를 구하기 위해 풀어야 하는 QP 문제는 P, q, A, b 중에서 q와 b만이 매 스텝마다 값이 변하게 된다. 이 논문에서는 이러한 특징을 활용한 알고리즘을 제시함으로써 예측제어를 비교적 빠른 시스템에도 실시간 적용할 수 있을 정도의 계산 성능을 얻는 방법을 제안한다.

IV. QP 해법을 이용한 예측제어의 C-code 구현

앞서 기술한 QP 문제의 해법은 NNLS의 풀이뿐 아니라 특이값 분해와 같은 복잡한 계산이 필요하다. 하지만 선형 시불변 시스템에 대한 예측제어를 구현하기 위해 매 스텝마다 풀이되어야 하는 QP 문제에서는 P와 A가 고정되어 있고 q와 b만 매 스텝마다 변하는 특징을 가진다. 따라서 이러한 특징을 활용하면 계산 효율을 크게 상승시킬 수 있는 알고리즘 구현이 가능하다. 매 스텝마다 값이 바뀌는 변수들을 나타내기 위해 아래첨자를 사용하기로 한다.

P가 고정되기 때문에 P에 대한 특이값 분해, 그리고 $\sqrt{P}$, P^{-1} 등의 계산은 offline으로 계산해도 된다. 예측제어 알고리즘에서 기본적인 사칙연산, 행렬간의 곱, 행렬과 벡터의 곱, 벡터의 덧셈 및 뺄셈, 그리고 NNLS 등의 계산만이 online으로 계산되어져야 한다. 그림 1은 이러한 내용을 반영하여 제안한 예측제어 구현 알고리즘의 흐름도를 나타낸다.

흐름도에서 offline 계산 부분을 마이크로콘트롤러에서 수행한다면 특이값 분해에 대한 C-code 기반의 풀이법이 필요하다. 행렬에 대한 특이값 분해를 위한 C-code 알고리즘은 [12]에서 찾아볼 수 있다. NNLS의 풀이 알고리즘은 [9]에 제시되고 있으며 이에 대한 C-code는 [1]과 같이 인터넷을 통해 구할 수 있다.

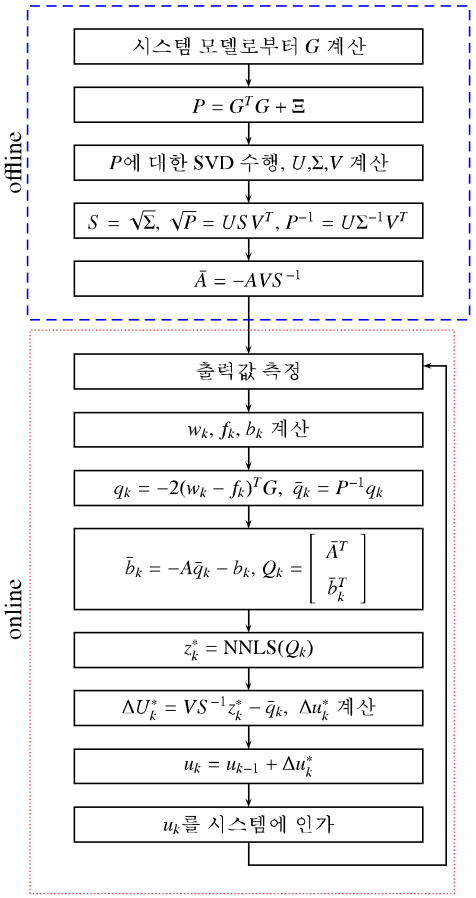


그림 1. 제안된 예측제어 알고리즘.

Fig. 1. Proposed algorithm for predictive control.

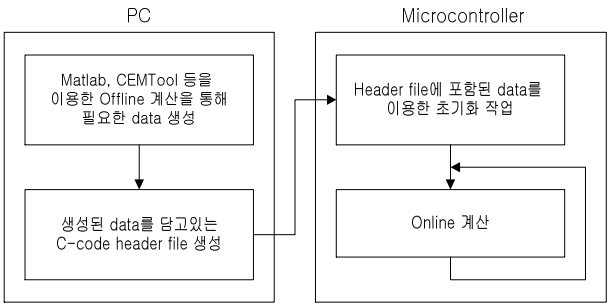


그림 2. 예측제어 구현 흐름도.

Fig. 2. Flow chart for the implementation of predictive control.

만약에 offline 계산부분을 마이크로콘트롤러에서 수행하는 것이 아니라 Matlab이나 CEMTool과 같은 소프트웨어를 이용하여 처리할 경우에는 역행렬, 특이값 분해 등의 계산이 지원되는 함수를 통해 손쉽게 계산된다. 따라서 offline 계산을 모두 수행한 후에 online 계산에서 활용되는 data들을 fprintf 함수를 활용하여 C-code header file에 출력하고 이것을 마이크로콘트롤러에서 수행될 application에서 사용하는 방식을 사용한다면 NNLS의 풀이에 대한 해만 C-code로 구현하면 예측제어 알고리즘을 마이크로콘트롤러에서 수행될 수 있도록 구현이 가능하다. 그림 2는 이러한 내용을 정리하여 담고 있는 개념도이다. offline 계산에 대한 부분은 PC 상에서 수행되고 online 계산부분은 마이크로콘트롤러에서 수행된다.

V. 수치예제

예제 1: P, q, A, b가 식 (8)과 같이 주어지는 QP 문제를 대상으로 아래의 (i)에서 (v)까지 다섯가지 경우에 대하여 수치실험을 수행하였다.

$$P = \begin{bmatrix} 0.7406 & 1.0186 & 0.8131 \\ 1.0186 & 1.8841 & 1.0338 \\ 0.8131 & 1.0338 & 0.9265 \end{bmatrix}, \; q = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix},$$

$$A = \begin{bmatrix} 0.0979 & -0.2690 & 0.1038 \\ -0.7100 & -0.0329 & -0.5369 \\ -0.2785 & 0.1960 & -0.1515 \\ -0.0424 & -0.5746 & 0.0073 \\ -0.0342 & -0.0212 & 0.0336 \\ -036749 & 0.1376 & -0.4923 \end{bmatrix}, \; b = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{8}$$

(i) Matlab의 'quadprog' 명령어를 이용한 풀이

(ii) II 절에서 제안된 방식을 C-code로 구현한 것을 PC 상에서 풀이

(iii) II 절에서 제안된 방식을 C-code로 구현한 것을 DSP 상에서 풀이

(iv) IV 절에서 제안된 방식을 C-code로 구현한 것을 PC 상에서 풀이

(v) IV 절에서 제안된 방식을 C-code로 구현한 것을 DSP 상에서 풀이

계산에 사용된 PC는 CPU 클럭속도가 2.3GHz인 Intel Core i5 2410M 프로세서를 기반으로 한 PC를 사용하였다. DSP는 Texas Instrument사의 TMS320F28335를 이용하였다. TMS320F28335는 클럭속도가 150MHz이며 floating point 연산을 하드웨어적으로 지원하는 특징을 가지고 있다. 프로그램 메모리로 512KB의 플래쉬 메모리를 가지며 68KB의 RAM을 가진다. 계산에 소요된 시간은 주어진 QP 문제를 5000회를 풀이할 때 소요된 시간 5000으로 나누는 방식을 사용하여 측정하였다. 여기서 (iv)와 (v)에서 IV 절에서 제안된 방식을 사용했다고 했는데, 부연하자면 P와 A는 고정되었다고 가정하고 $S, P^{-1}, \bar{A}$ 등을 미리 계산하여 5000회를 풀이하는 동안에는 다시 계산하지 않는 방식을 사용하였다는 것이다. 표 1은 각 방법별로 QP 풀이에 걸린 소요시간을 비교하고 있다. Matlab을 이용하였을 때 보다 C-code로 구현한 풀이법을 PC 상에서 수행했을 때 풀이 속도가 월등하게 나타나는 것을 볼 수 있다. DSP 상에서 수행했을 경우 Matlab에서 수행한 것보다 느린 것으로 나타나지만 이것은 PC와 DSP의 계산 성능차이가 큰 것에 기인한다. IV 절에서 제안된 방식을 이용하여 C-code를 이용한 QP 풀이법을 구현하였을 경우에는 PC와 DSP 상에서 공통적으로 큰 속도 개선 효과가 나타나는 것을 볼 수 있다. 제안된 방식을 이용했을 경우 DSP 상에서의 소요시간이 PC 상에서 Matlab으로 풀이했을 때의 소요시간보다도 짧아지는 것을 확인할 수 있었다. 이러한 속도개선 효과는 제안된 방식을 마이크로콘트롤러에 탑재하여 실제 시스템을 제어할 때 실시간성을 확보하는데 중요한 역할을 할 것이다.

표 1. QP 계산 속도의 비교.

Table 1. Speed comparison for computing QP.

방법	소요시간
(i)	$242us$
(ii)	$20us$
(iii)	$733us$
(iv)	$2.82us$
(v)	$126us$

예제 2: 이 논문에서 제안된 방법은 QP를 기반으로 한 모든 예측제어기법에 적용될 수 있다. 한 예로써 여러 가지 예측제어기법 중 DMC (Dynamic Matrix Control)을 적용하기로 한다. DMC는 시스템의 계단응답(step response)을 이용하여 예측자를 구성하는 예측제어기법으로서 안정한 시스템 또는 적분기를 포함한 시스템에 적용할 수 있는 특징이 있다. DMC의 경우 식 (5)에 주어진 예측자를 구성하는데 있어서 G와 f_k가 각각 다음과 같이 주어진다.

$$G=\begin{bmatrix} g_1 & 0 & \cdots & 0 \\ g_2 & g_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_{N_c} & g_{N_c-1} & \cdots & g_1 \\ \vdots & \vdots & \ddots & \vdots \\ g_{N_p} & g_{N_p-1} & \cdots & g_{N_p-N_c+1} \end{bmatrix},$$

$$f_k=\begin{bmatrix} \bar{y}_k \\ \bar{y}_k \\ \vdots \\ \bar{y}_k \end{bmatrix}+\begin{bmatrix} g_2-g_1 & g_3-g_2 & \cdots & g_N-g_{N-1} \\ g_3-g_1 & g_4-g_2 & \cdots & g_{N+1}-g_{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{N_p+1} & g_{N_p+2}-g_2 & \cdots & g_{N_p+N-1}-g_{N-1} \end{bmatrix}\begin{bmatrix} \triangle u_{k-1} \\ \triangle u_{k-2} \\ \vdots \\ \triangle u_{k-(N-1)} \end{bmatrix}$$

$$+\begin{bmatrix} h \\ 2h \\ \vdots \\ N_p h \end{bmatrix}u_{k-N}$$

여기서 $\bar{y}_k$는 출력에 대한 측정값, g_i는 $k=0$인 시점에 계단입력을 시스템에 인가했을 때 $k=i$인 시점에 나타나는 계단응답에 대한 정보로써 n_u개의 입력과 n_y개의 출력을 갖는 다변수 시스템의 경우 $n_y \times n_u$의 차원을 갖는 행렬이다[11]. 시스템이 안정한 시스템이거나 적분기를 포함한 시스템일 경우 계단응답이 정상상태에 도달하는데 소요되는 시간을 N을 찾을 수 있다. 적분기를 포함한 시스템에서 계단응답이 정상상태에 이른다는 것은 출력이 일정한 기울기를 갖는 형태로 나타나는 것을 의미하다. 시스템의 계단응답이 정상상태에 이르게 되면 $h=g_{i+1}-g_i,\ i \geq N$이 성립한다. 안정한 시스템의 경우 $h=0$이고 적분기를 포함한 시스템의 경우 $h \neq 0$이다.

DMC를 적용하기 위한 대상 시스템으로 다음과 같은 전달함수를 갖는 distillation column 시스템을 사용하였다[6].

$$G(s)=\begin{bmatrix} \frac{3.04}{s} & \frac{-278.28}{s(s^2+36s+180)} \\ \frac{0.052}{s} & \frac{319.47}{s(s^2+36s+180)} \end{bmatrix} \quad (9)$$

입력 u_1과 u_2는 $-1 \leq u_1 \leq 1,\ -1 \leq u_2 \leq 1$의 제약조건을 가진다고 가정하였다. 시스템의 단위계단 응답을 포함하여

표 2. 예측제어 연산 소요시간.

Table 2. Computation time taken for predictive control.

N_c	N_p	보통 연산 시간 $[ms]$	최장 연산 시간 $[ms]$
2	3	0.47	0.67
4	6	0.92	1.86
6	9	1.55	3.40
8	12	2.33	7.15
10	15	3.32	12.17

offline 계산들은 Matlab을 이용하여 수행하였으며 IV 절에서 제안된 방식을 이용하여 (9)에 대한 예측제어 실험을 C-code로 구현하여 DSP 상에서 수행하였다. 이 시스템은 적분기를 포함하는 시스템이기 때문에 h는 영행렬이 아닌 값을 갖는다. 주기적인 시행을 위하여 DSP의 타이머 인터럽트를 사용하였으며 인터럽트 주기는 50ms로 설정하였다. 연산에 소요된 시간의 측정은 오실로스코프를 이용하였다. 이를 위해 타이머 인터럽트 서비스 내에서 연산이 시작되는 시점에 DSP의 특정 핀에 논리값 1을 출력하고 연산이 모두 끝난 시점에 동일 핀에 논리값 0를 출력하였다. 여러 가지 제어구간 및 예측구간에 대해 예측제어기를 DSP상에서 수행한 결과 표 2와 같은 연산소요시간에 대한 실험 결과를 얻었다. 여기서 $N=40$으로 공통이다. 표에서 최장 연산시간은 특정 제어구간 및 예측구간을 사용했을 경우 연산에 소요되는 가장 긴 시간을 나타낸다. 예측제어의 해가 제약조건에 걸쳐있지 않은 경우에는 보통 연산시간이, 그리고 제약조건에 걸쳐있는 경우에는 최장 연산시간이 소요되었다. 제어구간과 예측구간을 증가시킬수록 연산에 소요되는 시간이 길어지지만 인터럽트 주기인 50ms 이내에 연산이 완료되어 실시간 수행이 가능하였다. 하지만 제어구간과 예측구간을 계속해서 증가시킬 경우 DSP가 가지는 RAM의 한계로 인해 링커오류가 발생하므로 이를 피하는 한도 내에서 구간을 선정하여야 했다. 이 예제의 경우 제어구간과 예측구간을 같게 설정하고 19이상의 제어구간을 사용했을 때부터 링커오류가 발생하였다.

상기의 실험을 통해 제안되는 방법을 이용하여 마이크로콘트롤러상에서 비교적 빠른 샘플타임의 경우에도 실시간 연산이 가능한 예측제어 알고리즘을 구현할 수 있음을 살펴보았다. 물론 예측구간이나 제어구간을 매우 큰 경우나 샘플타임이 매우 빠를 경우에는 실시간 성능을 만족할 수 없을 것이다. 하지만 제안된 방법이 예측제어를 마이크로콘트롤러에 탑재하여 실제 구현하기 위해 필요한 토대로써 사용되기에는 충분할 것으로 판단된다.

VI. 결론

QP 문제로 설정되는 예측제어 알고리즘을 마이크로콘트롤러에 탑재하여 실제 적용하는데 있어서 가장 큰 문제중 하나는 C-code로 구현된 QP 문제의 해법을 손쉽게 구할 수 없다는 것이다. 이 논문에서는 예측제어 알고리즘을 실제 마이크로콘트롤러 시스템에 탑재하여 적용해 볼 수 있도록 C-code로 작성가능한 QP 문제의 해법을 제안하고 실시간 계산 성능을 향상시키기 위한 방법을 제안하였다. 제안된

방법을 이용하여 C-code로 구현한 예측제어 알고리즘을 DSP 상에서 수행함으로써 실시간 계산 가능성을 살펴보았다. 실험을 통해서 제안된 방법을 이용할 경우 비교적 샘플 타임이 빠른 시스템에도 실시간 계산이 가능한 것을 확인하였다. 제안된 방법을 이용하면 예측제어 알고리즘을 마이크로콘트롤러 시스템에 탑재 가능하도록 구현할 수 있어 예측제어를 단순히 Matlab과 같은 소프트웨어를 이용한 모의실험에 그치는 것이 아니라 실제 시스템의 제어에도 적극적으로 활용할 수 있을 것으로 기대된다.

REFERENCES

[1] http://hesperia.gsfc.nasa.gov/~schmahl/nnls/nnls.c

[2] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulus, "The explicit solution of model predictive control via multiparametric quadratic programming," *Proc. of American Control Conference*, pp. 872-876, Chicago, Illinois, 2000.

[3] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, "Generalize predictive control-part1. the basic algorithm," *Automatica*, vol. 23, no. 9, pp. 137-148, 1987.

[4] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, "Generalized predictive control-part2. Extensions and interpretations," *Automatica*, vol. 23, no. 2, pp. 149-160, 1987.

[5] C. E. Garacia and A. M. Morshedi, "Quadratic programming solution of dynamic matrix control (QDMC)," *Chemical Engineering Communications*, vol. 46, pp. 73-87, 1986.

[6] A. N. Gundes, H. Ozbay, and A. B. Ozguler, "Pid controller synthesis for a class of unstable mimo plants with i/o delays," *Automatica*, vol. 43, no. 1, pp. 135-142, 2007.

[7] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361-1379, 1996.

[8] K. I. Kouramas, N. P. Faisca, C. Panos, and E. N. Pistikopoulos, "Explicit/multi-parametric MPC (Model Predictive Control) of linear discrete-time systems by dynamic and multi-parametric programming," *Automatica*, vol. 47, pp. 1638-1645, 2011.

[9] C. L. Lawson and R. J. Hanson, Solving Least Squares Problems, Prentice-Hall, Englewood Cliffs, New Jersey, 1974.

[10] J. H. Lee, M. Morari, and C. E. Garcia, "State-spave interpretation of model predictive control," *Automatica*, vol. 30, no. 4, pp. 707-717, 1994.

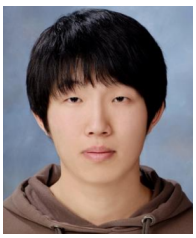
[11] P. Lundstrom, J. H. Lee, M. Morari, and S. Skogestad, "Limitations of dynamic matrix control," *Computers Chemical Engineering*, vol. 19, no. 4, pp. 409-421, 1995.

[12] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Plannery, "Numerical recipies in C," Cambridge University Press, Cambridge, 1988.

[13] P. Trondel, T. A. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit mpc solutions." *Automatica*, vol. 39, no. 10, pp. 489-497, 2003.

이 영 삼

제어 · 로봇 · 시스템학회 논문지 제15권 제4호 참조.



경 기 영

2014년 인하대학교 전기공학부 졸업. 2014년~현재 인하대학교 대학원 전기과 석사과정 재학중. 관심분야는 제어기 구현 및 모델 분석.

박 재 헌

2014년 인하대학교 전기공학부 졸업. 2014년~현재 인하대학교 대학원 전기공학과 석사과정 재학중. 관심분야는 예측제어.