Journal of Institute of Control, Robotics and Systems (2015) 21(9):827-835
http://dx.doi.org/10.5302/J.ICROS.2015.15.0086 ISSN:1976-5622 eISSN:2233-4335

RCP 시스템에서 사용가능한 DMC (Dynamic Matrix Control) 블록의 개발과 응용

Development of a DMC Block for Use with an RCP System and its Application

이 영 삼*, 유 광 명
(Young-Sam Lee[1,*] and Kwang-Myung Yu[2])
[1]Department of Electrical Engineering, Inha University
[2]Power Generation Lab, KEPCO Research Institute

Abstract: In this paper, we present the implementation method of Dynamic Matrix Control(DMC) block for use with a Rapid Control Prototyping(RCP) system and consider the speed control of a DC motor using the developed DMC block. Firstly, we briefly introduce a lab-built RCP system. Secondly, we present a method for implementing a DMC block using C-language, which enables the DMC algorithm to be represented in a library block that can be used in a Simulink environment. Finally, we use the developed DMC block for the speed control of a DC motor, through which we show that the DMC-based control system can be easily implemented and applied to the real-time control of systems with relatively fast dynamics.

Keywords: dynamic matrix control, rapid control prototyping, DC motor control, realtime control

I. 서론

예측제어는 특정형태의 제어입력을 제어의 대상이 되는 시스템에 인가했을 때 발생할 미래의 출력을 모델정보를 이용하여 예측하고 시스템에 인가할 수 있는 여러가지의 제어입력 중에서 비용함수를 최소로 만드는 제어입력을 찾아 시스템에 적용하는 제어기법이다. 예측제어에서는 제어입력을 계산하기 위해 매 샘플마다 이러한 최적화 문제를 풀어야 하기 때문에 연산량이 많이 소요되는 제어기법에 속한다.

예측제어는 사용하는 모델의 종류에 따라 상태공간 모델을 이용한 예측제어[1], 전달함수 모델을 이용한 예측제어[2,3], 시스템의 단위계단 응답을 이용하는 예측제어[4] 등 다양한 형태의 기법들로 구분된다. 상태공간 모델 및 전달함수 모델과 같은 수학적 모델을 이용하는 예측제어기법과 달리 DMC (Dynamic Matrix Control)는 단위계단 입력에 대한 시스템의 응답을 이용하여 미래를 예측하기 위한 예측자(predictor)를 구성할 수 있다는 점에서 실용성이 높은 제어기법이라 할 수 있다.

예측제어는 시스템이 가지는 입/출력 제약조건을 체계적으로 고려할 수 있다는 점에서 다른 제어기법과 차별화된다. 또한 다변수 시스템으로의 확장이 용이하다는 장점도 가진다. 예측제어가 제약조건을 고려할 수 있는 것은 제어량을 계산하는 문제가 제약조건을 갖는 최적화 문제로 설정되기 때문에 가능한 것인데 여러가지 최적화문제중 주로 QP (Quadratic Programming) 문제로 귀결된다. QP 문제의 해는 비교적 복잡한 과정을 거쳐서 얻어지기 때문에 시간이 많이 소요되고 이로 인해 예측제어는 화학공정과 같이 비교적 느린 시스템에 적용되어왔다. Matlab과 같은 software에서는 QP 문제의 해를 구할 수 있는 명령어를 제공한다.

예측제어의 차별화된 장점에도 불구하고 예측제어를 실제 시스템에 적용하는 사례는 생각보다 많지 않다. Matlab에는 QP 문제의 해법이 명령어로 제공되기 때문에 이것을 이용하여 예측제어에 대한 모의실험은 수행해 볼 수 있다. 하지만 실제 시스템에 대한 제어실험을 수행하기 위해서는 예측제어 알고리즘을 마이크로콘트롤러에 탑재가능한 C-code로 구현해야 하는데 이 과정이 쉽지 않다. QP를 풀이할 수 있는 C-code 기반의 해를 손쉽게 구할 수 없는 것이 가장 큰 이유이다. 만약 QP 문제의 해법에 대한 C-code 기반의 알고리즘을 손쉽게 구현할 수만 있다면 예측제어를 실제 시스템에 적용하는 연구가 지금보다는 훨씬 더 활발해 질 것으로 생각된다.

최근에는 제어 시스템을 구성하는데 있어 수작업에 의한 coding 방식대신 RCP 시스템을 이용하는 접근 방식이 많이 사용되고 있다. 대표적인 RCP 시스템으로는 Matlab/Simulink를 기반으로 한 RCP 시스템을 들 수 있는데, RCP 시스템을 이용하게 되면 제공되는 기능블럭을 조합하여 제어기 모델을 설계하고 C-code 생성기능을 이용하여 마이크

* Corresponding Author
Manuscript received May 27, 2015 / revised June 10, 2015 / accepted June 29, 2015
이영삼: 인하대학교 전기공학과(lys@inha.ac.kr)
유광명: 한전 전력연구원 발전연구소(kmyu@kepco.co.kr)
※ 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 ICT융합 고급인력과정지원사업의 연구결과로 수행되었음(IITP-2015-H8601-15-1003).

Copyright© ICROS 2015

로콘트롤러에 탑재가능한 제어알고리즘을 생성해 준다. code 생성에 의한 방식을 이용하지 않고 설계된 Simulink model 자체가 실시간 제어기 역할을 수행하는 구조의 RCP도 있다[5]. RCP 환경을 이용할 경우 블럭다이어그램을 통한 제어기 개발작업이 가능하고 입/출력 부분을 처리하기 위한 고민없이 제어 알고리즘 자체의 개발에만 집중할 수 있는 장점이 있다.

이 논문은 시스템의 단위계단 응답을 모델정보로 사용하는 예측제어 기법인 DMC를 Simulink 기반의 RCP 시스템내에서 사용할 수 있도록 기능블럭으로 구현하는 방법을 제시하고 이를 통해 DMC를 기반으로 한 예측제어시스템을 손쉽게 구성할 수 있는 방법을 제안한다. 제안되는 블럭은 C-code를 기반으로 구현되므로 고속 연산이 가능하고 그 결과 비교적 응답이 빠른 시스템에도 실시간 적용이 가능한 장점을 가진다. 제안되는 DMC 블럭은 전통적인 RCP 환경내에서 사용하여 마이크로콘트롤러에 탑재할 수 있도록 C-code로도 생성할 수 있다. 이 논문에서는 연구실에서 개발한 RCP 환경내에서 제안되는 DMC 블럭을 사용할 것이다.

본 논문은 다음과 같이 구성된다. 먼저 II 장에서는 연구실에서 개발된 RCP 시스템에 대해 소개한다. III 장에서는 DMC 블럭을 구현하기 위한 방법을 제안한다. IV 장에서는 DC 모터를 대상 시스템으로 DMC를 적용하는 절차적 방법을 정리하고 속도 제어를 DMC로 구현한다. 마지막으로 V 장에서는 논문에서 제시된 결과들에 대한 결론을 맺는다.

II. DAQ RCP의 소개

Rapid Control Prototyping (RCP) 시스템이란 제어시스템 개발자가 단기간에 제어 알고리즘을 효율적으로 설계, 개발, 검증하기 위해 사용하는 개발환경을 말한다. 전통적으로 RCP 시스템은 Simulink와 같은 블럭다이어그램 방식의 모델링 프로그램, 하드웨어 입/출력 장치를 제어하기 위해 사용하는 라이브러리 블럭, C-code 생성기, 실시간 타겟(Target) 컴퓨터, 타겟 컴퓨터와 통신하는 호스트(Host) 컴퓨터 등으로 구성된다. Simulink를 이용하여 제어기를 설계하고 모의실험등을 통해 테스트 해본 후 만족스러운 결과가 나올 경우 입/출력 제어 라이브러리 블럭을 사용하여 실제 시스템을 제어 할 수 있도록 제어기를 구성한다. 자동 C-code 생성기를 이용하여 구성된 제어기 모델에 대한 C-code를 생성하고 컴파일 한 후 실시간 타겟 컴퓨터에 다운로드하여 실시간 제어 실험을 수행하게 된다. 하드웨어 입/출력 장치를 제어하기 위한 Simulink용 라이브러리 블럭이 제공되고 C-code 생성 기능이 있기 때문에 제어 시스템 개발자는 지루하고 오류가 발생하기 쉬운 수작업 코딩(Coding)을 하지 않아도 되고 오로지 제어 알고리즘의 개발이라는 본연의 목적에만 집중할 수 있게 된다.

RCP 시스템과 관련된 몇몇 소프트웨어가 상업적으로 판매되고 있는데 Matlab/Simulink는 가장 잘 알려지고 널리 사용되는 프로그램이다. Simulink의 추가 제품인 Real-time Workshop (RTW)은 Simulink로 구성한 블럭 다이어그램 모델에 대해 자동으로 ANSI-C Code를 생성해 준다[6]. Simulink의 또다른 추가제품인 Embedded Coder는 특정 내

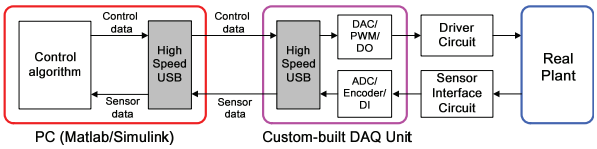

그림 1. 연구실에서 개발된 DAQ RCP 시스템의 개념도.

Fig. 1. The concept diagram of a lab-built DAQ RCP system.

장형 프로세서에 특화된 C-code를 생성해 줌으로써 제품 생산 기간을 단축해 주는 기능을 제공해준다[7]. Matlab/Simulink와 RTW는 개방형 구조를 취함으로써 직접 개발한 하드웨어 장치에 대해서도 RCP 소프트웨어를 적용할 수 있도록 하고 있다. 이런 개방성 때문에 자체개발된 하드웨어에 Matlab/Simulink/RTW를 사용한 RCP 시스템들이 학술적인 연구로써 다수 제안되었다[5,8-12]. 더불어 RCP 시스템의 적용에 대한 연구들도 다수 발표되고 있다[13,14].

이 논문에서는 [5]에서 저자가 제안하였던 RCP 시스템을 사용한다. [5]에서 제안된 RCP는 high speed USB 통신기능을 갖춘 DAQ(Data Acuisition) 보드와 Matlab/Simulink를 기반으로 한다. DAQ 보드를 기반으로 하고 있으므로 이 논문에서는 편의상 DAQ RCP로 부르기로 한다. DAQ RCP 시스템은 그림 1과 같이 플랜트를 제외하고 크게 2개의 하부 시스템(subsystem)으로 구성된다. 첫번째 하부 시스템은 MS Windows 상에서 Matlab/Simulink가 실행되는 PC 시스템이고 high speed USB 통신기능을 갖춘 DAQ 장치가 두번째 하부 시스템을 구성한다. PC와 DAQ 장치는 high speed USB 통신을 이용하여 데이타(제어 데이타 및 센서 데이타)를 주고 받는다. Simulink를 이용하여 구성된 제어기 모델에 대해 C-code를 자동생성하고 컴파일하여 내장형 제어장치(Embedded control unit)에 다운로드하여 제어기를 실행시키는 기존의 RCP 시스템과 달리 DAQ RCP 시스템은 C-code 생성과정없이 바로 Simulink가 제어기 역할을 수행하는 것이 특징이다. 센서데이터의 측정과 제어 데이터를 적용하는 것은 DAQ 장치가 관장한다. 그림 2는 DAQ RCP 시스템에서 제어가 수행되는 방식을 흐름도로 나타낸 것이고 각 단계를 정리하면 다음과 같다.

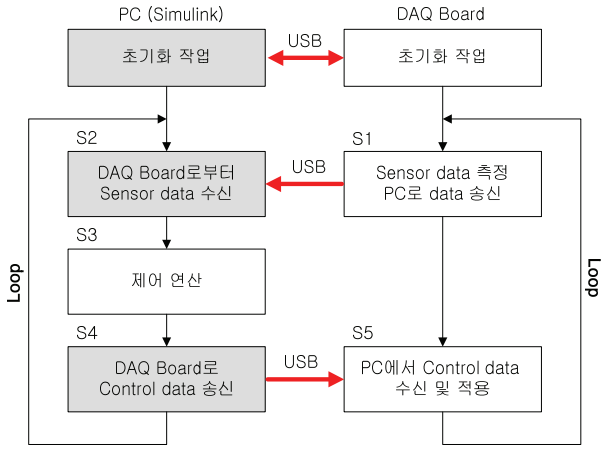

그림 2. [5]에서 제안된 DAQ RCP 시스템의 수행 흐름도.

Fig. 2. The flowchart of the DAQ RCP system proposed in [5].

S1: DAQ 장치가 제어에 필요한 센서 데이타를 측정한 후 USB 통신을 통해 PC 측으로 전송한다.
S2: PC에서 수행중인 Simulink가 DAQ 장치에서 전송된 센서 데이터를 수신한다.
S3: 수신된 센서 데이터를 기반으로 제어연산을 수행한다.
S4: 계산된 제어 연산 결과를 DAQ 장치로 전송한다.
S5: DAQ 장치에서는 PC 측에서 전송된 제어 데이타를 출력장치(DAC, PWM 등등)를 통해 출력한다.

DAQ 장치에서 수행되는 일련의 작업들은 마이크로콘트롤러에 의해서 제어된다. [5]에서 언급되었듯이 DAQ RCP 시스템은 대략 2 KHz 정도의 샘플레이트를 갖는 제어시스템을 실시간으로 구현할 수 있기 때문에 학부 및 대학원 연구에서 다루어지는 많은 제어시스템들을 다룰 수 있다 [19]. 제공되는 입/출력 기능블럭을 이용하여 Simulink 내에서 하드웨어 입/출력 장치를 효과적으로 다룰 수 있도록 설계되었다. 대표적인 입/출력 기능블럭은 Encoder 블럭, Digital Input 블럭, ADC 블럭, PWM 블럭, Digital Output 블럭, DAC 블럭등이 있다.

III. DMC 블록의 구현

이 장에서는 C-code S-function을 이용하여 DMC 블럭을 구현하는 방법을 제시한다. S-function은 Simulink에서 지원되는 라이브러리 블럭에 없는 블럭을 사용자가 직접 정의할 때 사용하는 컴퓨터 언어로써 Matlab 스크립트 언어 또는 C 언어를 이용하여 기술하게 된다. 특히 C-code를 이용하여 블럭을 정의할 경우 블럭의 연산속도가 Matlab 스크립트 언어에 의해 기술된 S-function 블럭보다 월등하게 빠르며 RCP 환경내에서 사용할 경우 C-code 생성기능을 지원하는 장점을 가지게 된다. 이 논문에서는 연구실에서 자체적으로 개발한 DAQ RCP를 사용하기 때문에 C-code를 생성할 필요는 없지만 고속계산을 가능하게끔 하기 위해 DMC 블럭을 C-code로 S-function을 구현하도록 한다.

그림 3은 DMC를 이용한 제어 시스템의 개념도이다. 이 논문에서는 DMC 블럭을 C-code 기반의 S-function으로 구현함으로써 DMC를 Simulink 내에서 사용할 수 있도록 하고 나아가 RCP 환경내에서 사용할 수 있도록 하는 것을 목표로 한다. 이를 통해 주로 모의실험에만 국한되어 사용되었던 DMC 기반의 제어를 실제 시스템의 제어에 적용할 수 있게 하는 환경을 제공하는 것을 목표로 한다. 제안되는 DMC 블럭을 DAQ RCP 환경내에서 사용하여 DC 모터를 제어하는 제어시스템을 구성한다면 그림 4와 같이 구성할 수 있다. 모터의 속도를 측정하고 DC 모터 드라이버를

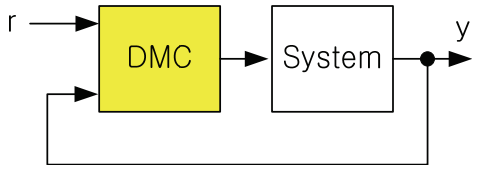

그림 3. DMC를 이용한 제어시스템의 블록선도.
Fig. 3. The block diagram of a DMC-based control system.

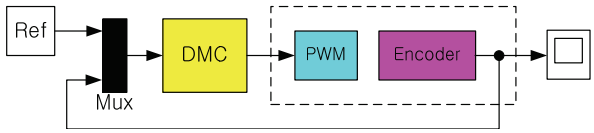

그림 4. DAQ RCP를 이용하여 Simulink로 구현한 DC 모터 제어 시스템의 블록선도.
Fig. 4. The block diagram of the DC motor control system implemented using the DAQ RCP in Simulink.

PWM 구동하기 위하여 DAQ RCP에서 제공되는 Encoder 블럭과 PWM 블럭을 사용하고 DMC 기반의 제어는 DMC 블럭을 이용하여 처리한다.

DMC 블럭을 구현하기 위해서는 먼저 Quadratic programming에 대한 해법을 구현해야 하고 이것을 기반으로 DMC 알고리즘 자체를 구현하고 마지막으로 S-function을 이용하여 DMC 블럭을 정의해야 한다.

1. Quadratic programming 해법의 구현

예측제어 알고리즘을 구현하는데 있어서 가장 핵심이 되는 알고리즘은 Quadratic programming(QP)이다. DMC도 예측제어의 일종인만큼 DMC 블럭을 C-code로 구현하기 위해서는 QP 해법의 C-code 구현이 필수적이다. 이 논문에서는 [15]에서 제안된 방법을 사용하기로 한다.

일반적인 QP 문제는 다음과 같이 기술된다.

$$\min_{x} \ \frac{1}{2}x^T Px + q^T x, \ \text{Subject to} \ Ax \le b. \tag{1}$$

$P>0$인 경우 위의 문제는 볼록 최적화문제(convex optimization)가 되어 전역해가 보장되고 해가 유일하게 존재한다. 예측제어에서 풀어야 하는 QP 문제는 $P>0$을 만족하므로 $P>0$를 만족하는 특수한 경우의 해법을 사용하기로 한다. [15]에서 보여졌듯이 $P>0$일 경우 식 (1)에 주어진 QP 문제는 일련의 등가변형을 통해서 최종적으로 다음과 같은 NNLS (Nonnegative least-squares) 문제를 풀어 그 해를 통해 구할 수 있다.

$$\text{NNLS}: \min_{d} \ \| Qd - r \|, \ \text{Subject to} \ d \ge 0. \tag{2}$$

여기서

$$Q = \begin{bmatrix} \bar{A}^T \\ \bar{b}^T \end{bmatrix}, \ r = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

이고 $\bar{A} = -AVS^{-1}$, $\bar{b} = -A\bar{q} - b$, $\bar{q} = P^{-1}q$이며 V와 S는 각각 $\sqrt{P}$를 특이값 분해하여 얻는 행렬로써 $\sqrt{P} = USV^T$를 만족한다. 식 (2)에 주어진 NNLS 문제의 해를 d^*라 하면

식 (1)에 주어진 원래의 QP 문제의 해 x^*는 다음과 같이 구할 수 있다.

$$x^* = VS^{-1}\left[\frac{\bar{A}^T d^*}{(1 - b^T d^*)}\right] - \bar{q}$$

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) NNLS의 풀이
(c) $\sqrt{P}$에 대한 특이값 분해
(d) P^{-1}를 계산하기 위한 역행렬 계산

이중에서 (a)에 대한 계산은 C-code로 쉽게 구현가능하다. (b)의 NNLS의 해는 [16]에서 제시된 알고리즘을 사용하여 구현할 수 있다. (c), (d)를 처리하기 위해서는 제곱근 행렬 계산, 특이값 계산, 역행렬 계산등 비교적 복잡한 알고리즘을 구현하여야 한다. 하지만 예측제어에서 풀어야 되는 QP 문제는 시스템의 모델이 바뀌지 않을 경우 P가 상수행렬이므로 (c)와 (d)를 계산하기 위한 C-code를 직접 구현할 필요없이 Matlab과 같은 software를 이용하여 (c), (d)를 계산하고 C-code로 된 알고리즘에서는 결과값 사용하는 방식을 취하기로 한다. C 언어에서는 행렬과 벡터를 배열(array)로 나타내므로 Matlab 연산의 결과값을 header file에 배열로 출력한다. 이때 Matlab 명령어인 fprintf를 이용할 수 있다.

2. DMC의 구현

DMC는 시스템의 단위계단 응답을 시스템의 모델정보로 사용하는 예측제어의 일종으로 매 스텝마다 다음과 같이 제약조건을 만족하면서 주어진 비용함수를 최소화하는 최적화 문제를 풀이하여 제어량을 계산한다.

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

$$\text{Subject to } u_{\min}\le u_{k+j}\le u_{\max} \tag{4}$$

$$\triangle u_{\min}\le \triangle u_{k+j}\le \triangle u_{\max}$$
$$j=0,1,\cdots,N_c-1.$$

변수에 사용된 아래첨자는 시점을 나타내며 k는 현재시점을 나타낸다. 입력을 u라고 한다면 u_{k+j}는 현재 시점으로부터 j 스텝 이후의 입력을 의미한다. 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 대각행렬로써 $\| \triangle u_{k+j}\|_{\Lambda}^2=\triangle u_{k+j}^T\Lambda\triangle u_{k+j}$를 나타낸다. DMC에서는 매 스텝마다 $k+N_c-1$까지 제어증분량을 계산하지만 사용되는 것은 $\triangle u_k$로써 실제 시스템에 사용되는 제어입력은

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

와 같이 구해진다. DMC에서는 현재시점부터 N_p 스텝 이후까지의 예측 출력을 다음과 같이 나타낸다.

$$\hat{Y}_k=G\triangle U_k+f_k \tag{5}$$

여기서

$$\hat{Y}_k=\left[\hat{y}_{k+1}^T\ \hat{y}_{k+2}^T\cdots\hat{y}_{k+N_p}^T\right]^T$$
$$\triangle U_k=\left[\triangle u_k^T\ \triangle u_{k+1}^T\cdots\triangle u_{k+N_c-1}^T\right]^T$$

이고 G는 시스템의 단위계단응답으로 구성된 예측행렬이고, f_k는 free response로써 과거의 data가 미래의 출력에 기여하는 양으로 아래와 같이 나타내어 진다.

$$G=\begin{bmatrix} g_1 & 0 & \cdots & 0\\ g_2 & g_1 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ g_{N_c} & g_{N_c-1} & \cdots & g_1\\ \vdots & \vdots & \ddots & \vdots\\ g_{N_p} & g_{N_p-1} & \cdots & g_{N_p-N_c+1}\end{bmatrix}, \tag{6}$$

$$f_k=\begin{bmatrix}\bar{y}_k\\ \bar{y}_k\\ \vdots\\ \bar{y}_k\end{bmatrix}+\begin{bmatrix} g_2-g_1 & g_3-g_2 & \cdots & g_N-g_{N-1}\\ g_3-g_1 & g_4-g_2 & \cdots & g_{N+1}-g_{N-1}\\ \vdots & \vdots & \ddots & \vdots\\ g_{N_p+1}-g_1 & g_{N_p+2}-g_2 & \cdots & g_{N_p+N-1}-g_{N-1}\end{bmatrix}\begin{bmatrix}\triangle u_{k-1}\\ \triangle u_{k-2}\\ \vdots\\ \triangle u_{k-(N-1)}\end{bmatrix}+\begin{bmatrix}h\\ 2h\\ \vdots\\ N_ph\end{bmatrix}u_{k-N} \tag{7}$$

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[17]. N은 시스템이 안정한 시스템이거나 적분기를 포함한 시스템일 경우 계단응답이 정상상태에 도달하는데 소요되는 시간을 나타낸다. 적분기를 포함한 시스템에서 계단응답이 정상상태에 이른다는 것은 출력이 일정한 기울기를 갖는 형태로 나타나는 것을 의미하다. 시스템의 계단응답이 정상상태에 이르게 되면 $h=g_{i+1}-g_i,\ i\ge N$이 성립한다. 안정한 시스템의 경우 $h=0$이고 적분기를 포함한 시스템의 경우 $h\ne 0$이다. 여기서 G에는 아래 첨자가 붙지 않았는데 이것은 매 스텝마다 G의 값이 변하지 않기 때문이다. 미래의 기준입력 벡터를

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

로 정의하면 [15]에서 기술되었듯이 DMC 문제는 $\triangle U_k$에 대한 QP 문제가 되며 이때 P, q, A, b는 다음과 같이 주어진다.

$$P=G^TG+\Xi,\ q=-(W_k-f_k)^TG,$$
$$A=\begin{bmatrix}I_D\\ -I_D\\ I_L\\ -I_L\end{bmatrix},\quad b=\begin{bmatrix}I_V u_{\max}\\ -I_V u_{\max}\\ I_V(u_{\max}-u_{k-1})\\ -I_V(u_{\max}-u_{k-1})\end{bmatrix} \tag{8}$$

여기서 Ξ, I_D, I_L, I_V는

$$\Xi=diag[\Lambda\,\Lambda\cdots\Lambda],\ I_D=\begin{bmatrix}I_m & 0 & \cdots & 0\\ 0 & I_m & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & I_m\end{bmatrix},$$
$$I_L=\begin{bmatrix}I_m & 0 & \cdots & 0\\ I_m & I_m & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ I_m & I_m & \cdots & I_m\end{bmatrix},\ I_V=\begin{bmatrix}I_m\\ I_m\\ \vdots\\ I_m\end{bmatrix}$$

와 같이 주어지며 예측구간 및 제어구간의 길이에 따라 적절한 차원을 가진다. 식 (8)에서 알 수 있듯이 DMC를 구하

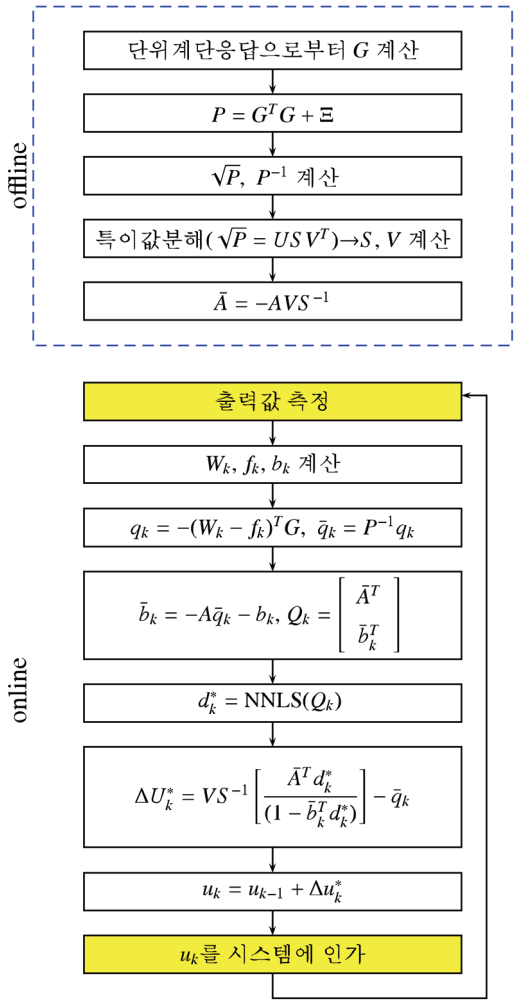

그림 5. 알고리즘 흐름도.

Fig. 5. The algorithm flowchart.

기 위해 매 스텝마다 풀어야 하는 QP 문제는 P, q, A, b 중에서 q와 b만이 매 스텝마다 값이 변하고 P와 A는 시스템 모델이 바뀌지 않는한 상수행렬이다. 따라서 P와 A와 관련된 값들은 Matlab을 이용한 offline 계산을 통해 미리 계산하여 결과를 구한 후 C-code용 header file에 출력해 놓고 q와 b를 생성하는 것은 online으로 계산되도록 C-code를 작성한다. q와 b를 q_k와 b_k로 나타내어 매 스텝마다 계산해야 하는 값임을 명시하기로 한다. 그림 5는 offline으로 계산되는 부분과 online으로 계산되는 부분에 대한 개략적인 흐름도를 나타내고 있다. 흐름도에 주어진 알고리즘을 C-code로 구현함으로써 DMC를 구현할 수 있다. C-code로 구현이 쉬운 행렬의 덧셈, 곱셈과 이미 해가 알려진 NNLS 등을 이용하여 알고리즘을 구현할 수 있음을 볼 수 있다.

3. DMC 블럭의 구현

앞서 제시된 방법을 통해 DMC를 C-code로 구현할 수 있다. 하지만 제어이론에 익숙하지 않은 학생이나 엔지니어들이 DMC와 같은 고급 제어기법에 대한 알고리즘을 직접 C로 구현하여 실제 시스템의 제어에 적용하는 것은 여전히 쉬운 일이 아니다. 만약 DMC 알고리즘이 Simulink에서 사용할 수 있는 블럭으로 제공된다면 사용자들은 블럭을 복사한 후 적당한 블럭 편집을 통해 DMC를 이용한 제어시스템을 구현할 수 있을 것이다. 본 연구에서는 이러한 사실에 착안하여 DMC 알고리즘을 Simulink에서 사용할 수 있는 블럭으로 구현하였다. Matlab에서는 사용자가 블럭을 정의하여 사용할 수 있게끔 S-function이라는 기능을 제공하고 있는데 S-function을 통해 블럭의 초기화(initialization) 함수, 입/출력 관계식 함수, 종료(termination) 함수등을 정의하게 된다. 그림 4의 개념도에서와 같이 DMC 블럭의 입력은 기준입력(reference)과 출력 데이터이고 출력은 DMC에 의해 구해진 제어량이 되도록 구현하였다. 블럭의 입력과 출력간의 관계식은 앞서 기술한 알고리즘을 근간으로 기술하였다.

IV. DMC 기반 DC 모터 속도 제어시스템 구현

이 장에서는 개발된 DMC 블럭을 DAQ RCP 환경내에서 사용하여 DC 모터의 속도를 실시간 제어하는 문제를 다룸으로써 제안되는 DMC 블럭을 통하여 손쉽게 제어시스템의 구현이 가능하고 더불어 응답이 빠른 시스템에도 DMC를 적용하는 것이 가능함을 예시한다. 실험에 사용할 DC 모터는 Maxon 사의 150W급 DC 모터이다. 모터에는 500 PPR 해상도의 Encoder가 달려있어 위치 및 속도를 측정할 수 있다. 그림 6은 제어실험에 사용된 실험 환경의 구성을 나타낸다. 그림에서 왼쪽부분은 Arduino Due를 기반으로 한 DAQ 보드, 가운데는 H-bridge를 이용한 DC 모터 드라이버, 그리고 오른쪽은 Maxon DC 모터이다. DAQ 보드는 USB를 통해 PC와 연결되어 있다. 모터드라이버의 H-brdige에 인가되는 PWM은 단극성 상보 PWM (Unipolar complementary PWM)을 사용하였다.

1. 단위계단 응답 데이터의 취득

DMC를 적용하기 위해서는 먼저 시스템의 단위계단 응답을 얻어내는 과정이 필요하다. 이를 위하여 DC 모터의 입력 전압으로 5V의 계단입력을 인가한 후 모터의 회전속도를 측정하였고 정규화(normalization)을 위하여 5로 나누어 주는 방법을 사용하였다. 그림 7은 단위계단 응답 실험을 하기 위해 구성한 Simulink 모델이다. 모델에서 Maxon Motor라고 이름붙은 블럭은 서브시스템으로써 그 내부는 그림 8과 같이 구성되어 있으며 DAQ RCP 환경에서 제공

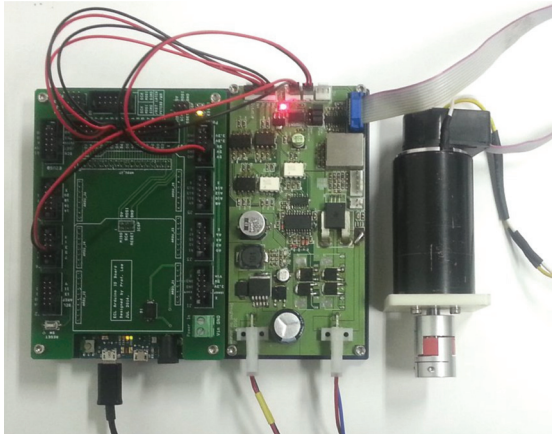

그림 6. DC 모터 속도제어 실험 세트.

Fig. 6. The experimental set for DC motor speed control.

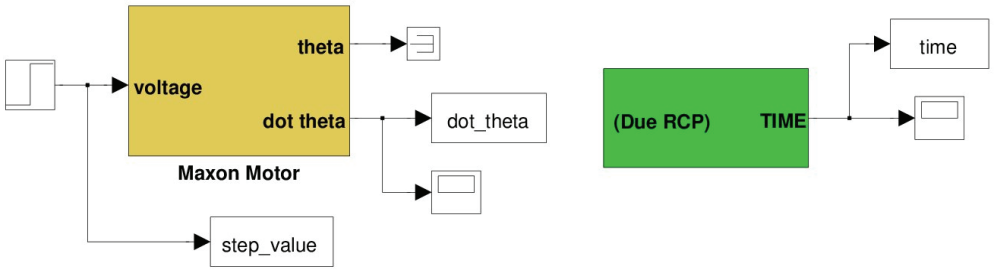

그림 7. 단위계단 응답실험을 위한 Simulink 모델.

Fig. 7. Simulink model for step response simulation.

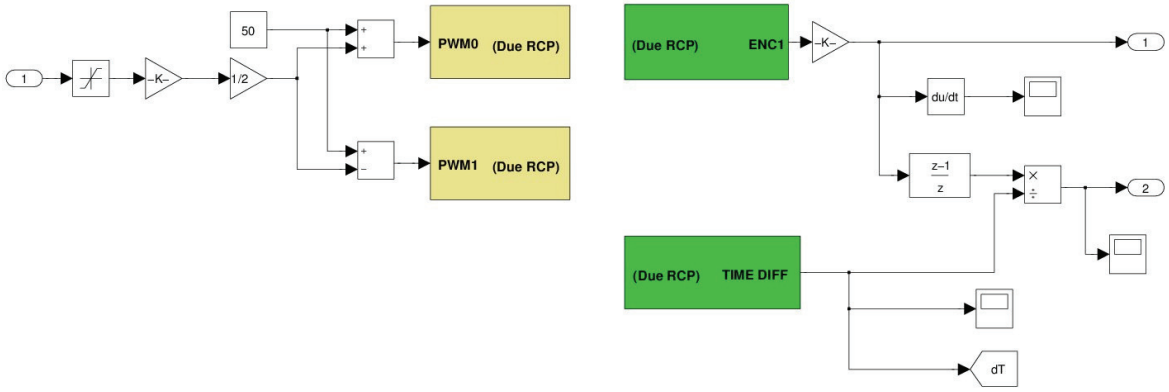

그림 8. DC 모터 서브 시스템.

Fig. 8. DC motor sub-system.

하는 입/출력 블럭인 PWM 블럭과 Encoder 블럭을 이용하여 모터드라이버에 PWM 신호를 인가하고 모터로부터 회전속도를 얻어내는 역할을 한다. Simulink 모델의 샘플링타임은 1 ms로 설정하였다.

단위 계단 응답 실험을 한번만 수행해서는 제대로 된 계단응답을 얻기가 곤란하다. 그림 9는 단위계단 응답 실험을 한번만 수행하고 얻어진 모터의 속도 궤적을 별다른 후처리 없이 나타낸 것이다. Encoder가 500 PPR(Pulse per revolution)의 해상도를 가지므로 이로 인해 정량화오차가 크게 나타나는 것을 볼 수 있다. 본 논문에서는 단위계단응답 실험을 30회 실행한 후 평균값을 사용하고 Matlab의 filtfilt 명령에 의한 추가적인 필터링을 적용하여 단위계단 응답이 부드러운 형태의 선으로 나타나도록 후처리하는 방법을 사용하였다. 그림 10은 이러한 후처리를 거쳐 얻은 단위계단 응답의 궤적을 나타낸다.

2. DMC를 이용한 속도제어

식 (7)의 N의 선정은 단위계단 응답이 완전히 정상상태에 이른 시점을 이용하여 정할 수 있다. 그림 10으로부터

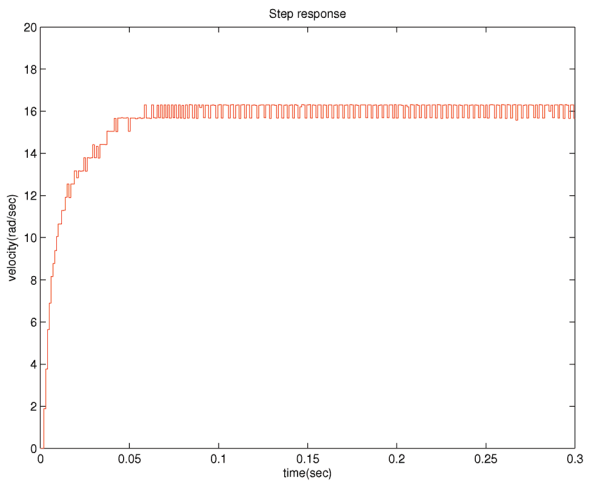

그림 9. 단위계단응답.

Fig. 9. Unit step response.

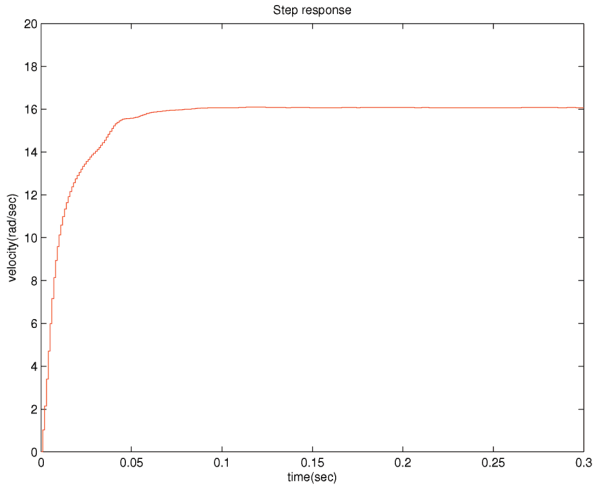

그림 10. 후처리를 거친 단위계단응답.

Fig. 10. Unit step response after post-processing.

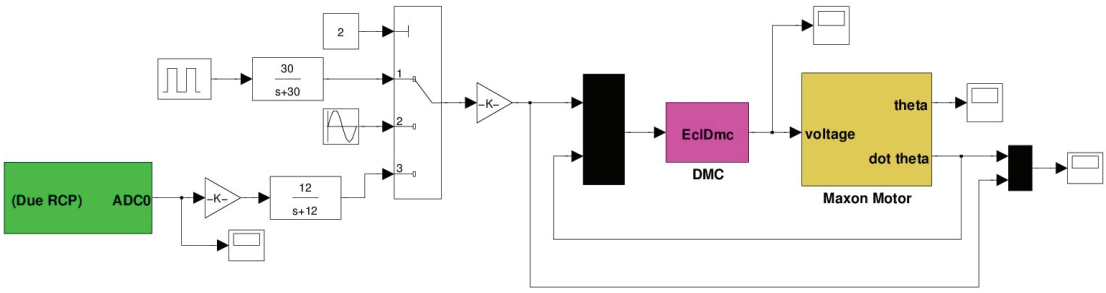

그림 11. DMC 블록을 이용하여 구성한 DC 모터의 속도 제어 시스템.

Fig. 11. DC motor speed control system constructed using a DMC block.

$t=0.3$이면 응답이 완전히 정상상태에 이른 것을 볼 수 있고 샘플링타임이 1ms 이므로 $N=0.3/0.001=300$으로 선정하였다. 제어구간과 예측구간은 각각 $N_c=3$, $N_p=3$으로 선정하였다. 전원공급기의 DC 공급 전압을 10V로 설정하였으므로 $u_{\max}=10$, $u_{\min}=-10$이다. 그림 11은 구현된 DMC 블럭을 이용하여 구성한 DC 모터의 속도제어 시스템에 대한 모델이다. 기준입력으로 구형파, 정현파, 그리고 가변저항을 통한 외부 입력등을 선택할 수 있도록 구성하였고 그림 11에서는 정현파 형태의 속도 기준입력이 사용되도록 설정된 것을 볼 수 있다.

그림 12는 정현파 형태의 속도기준입력을 DMC를 이용하여 추종하는 실험의 결과를 나타낸다. 파란색 선은 기준입력을 빨간색 선은 실제 모터의 속도를 나타낸다. 모터가 기준입력 속도를 충실히 추종하는 것을 볼 수 있다. 이 경우 DMC에 의해 생성된 제어량은 모두 제약조건의 범위 내에 위치한다(즉, 제약조건 비활성). 속도 기준입력의 진폭을 증가시켜 $-10\sim10$ volt의 전압으로는 주어진 기준입력을 추종할 수 없도록 해보았다. 그림 13은 이 경우의 추종성능을, 그림 14는 DMC에 의해 생성된 제어량의 크기를 나타낸다. 주어진 기준입력의 진폭이 커서 일부 구간에서는 추종이 제대로 되지 않는 것을 볼 수 있다. DMC에 의해 생성된 제어량은 주어진 제약조건 $-10\sim10$ volt를 정확하게 만족하는 것을 볼 수 있다. DMC에 의해 제어되는 DC 모터는 생성되는 제어량이 제약조건에 걸려있을 경우에는 추종이 완벽하게 되지 않지만 생성된 제어량이 제약조건 내에 있을 경우에는 기준입력을 충실하게 추종하는 것을 볼 수 있다.

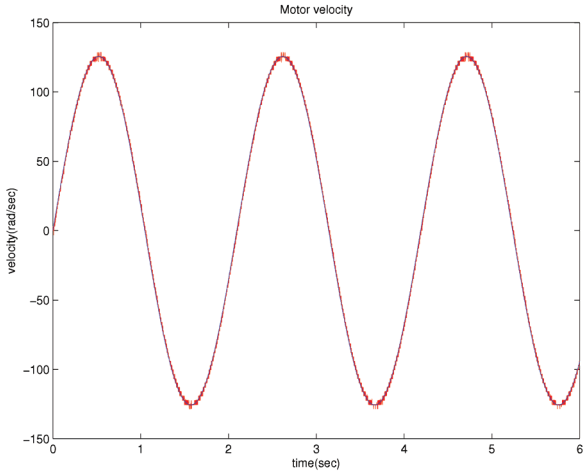

그림 12. 입력 제약조건이 비활성인 경우 DMC의 속도 추종 성능.

Fig. 12. The speed tracking performance of a DMC when the input constraint is not active.

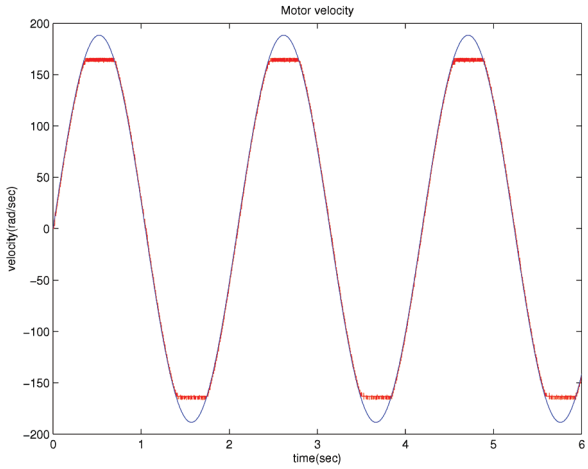

그림 13. 입력 제약조건이 활성일 경우 DMC의 속도 추종 성능.

Fig. 13. The speed tracking performance of a DMC when the input constraint is active.

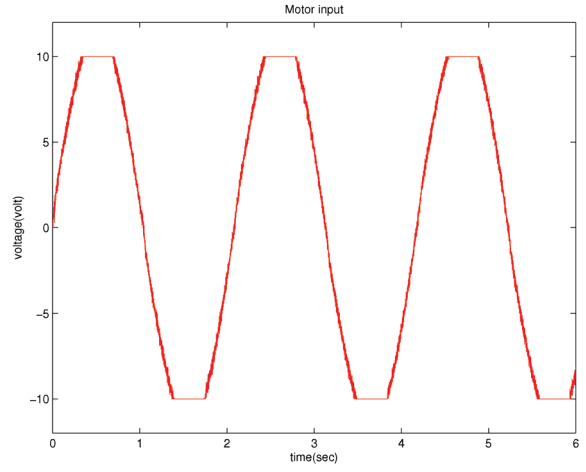

그림 14. DMC를 이용하여 생성한 제어량의 크기.

Fig. 14. The control value generated from a DMC.

모터의 속도는 고전적인 PI 제어기를 이용해서도 제어할 수 있다. 적절하게 튜닝된 PI 속도제어 시스템을 이용하여 위와 동일한 시험을 수행하여 보았다. 그림 15는 PI 제어시스템의 속도 추종 성능을, 그림 16은 PI 제어기에 의해 생성된 제어량의 크기를 나타낸다. PI 제어기의 경우 입력의 크기 제약조건을 고려할 수 없기 때문에 생성된 제어량이 제약조건을 크게 벗어나는 것을 그림 16에서 살펴 볼 수 있다. 이것은 결국 PI 제어기의 windup 현상으로 이어져 속도 추종 성능이 DMC의 경우보다 좋지 않은 것을 확인할 수 있다. 물론 PI 제어기도 anti-windup 기법을 이용할 경우 DMC와 유사한 성능을 얻을 수 있다[18]. 하지만 anti-windup 기법이 제약조건을 위반한 후에 조치가 취해지는 소극적 방법의 기법이라면 DMC는 예측을 통해 제약조건을 체계적으로 고려하는 적극적 방법이다. 이 논문에서 위와 같은 비교 실험을 수행한 것은 DMC가 제약조건을 조직적으로 고려할 수 있음을 명확하게 보여주기 위함이다. DMC

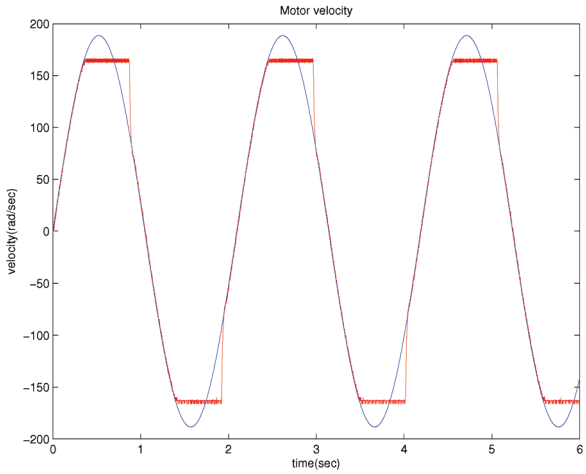

그림 15. 입력 제약조건이 활성일 경우 PI 제어의 속도 추종 성능.

Fig. 15. The speed tracking performance of a PI controller when the input constraint is active.

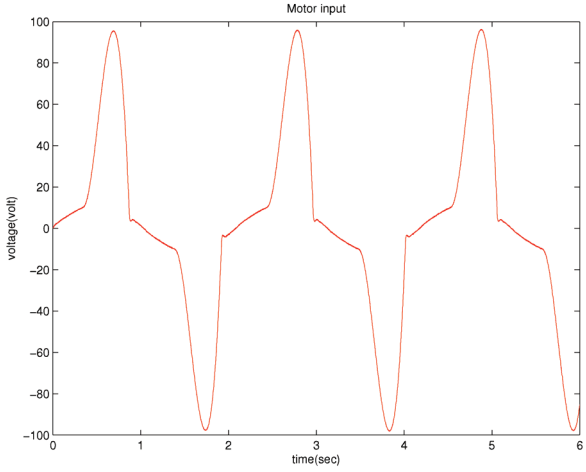

그림 16. PI제어를 이용하여 생성한 제어량의 크기.

Fig. 16. The control value generated from a PI controller.

를 이용한 모터제어 실험의 샘플링타임은 1ms, 즉 1초당 1000번의 연산을 수행하게 되는데 이것은 비교적 빠른 샘플링타임에 해당한다. 이를 통해 제안된 DMC 블럭을 이용할 경우 비교적 빠른 시스템에도 DMC를 적용할 수 있음을 알 수 있는데 이것이 가능한 것은 DMC 블럭이 C-code S-function으로 구현되었기 때문이다. 더불어 개발된 DMC 블럭을 RCP 환경하에서 사용함으로써 DMC와 같은 고급제어기법을 블럭의 복사 및 편집을 통해 손쉽게 구현할 수 있음을 알 수 있다.

V. 결론

이 논문에서는 Simulink를 기반으로 하는 RCP 시스템에서 사용할 수 있는 DMC (Dynamic matrix control) 블럭의 구현을 다루었고 개발된 DMC 블럭을 DC 모터의 속도 제어에 응용하는 문제를 다루었다. DMC 블럭의 구현을 위해 먼저 예측제어의 핵심이 되는 quadratic programming 문제에 대한 해법을 C로 구현하는 방법을 제안하였고 그 해법을 기반으로 DMC 알고리즘을 C-code로 구현하는 방법을 제시하였다. 제시된 방법을 기반으로 Simulink에서 사용할 수 있는 DMC 블럭을 S-Function을 이용하여 구현하였다. Simulink 상에서 하나의 기능블럭으로 DMC를 구현해 놓음으로써 해당 블럭을 RCP 환경과 같이 사용하여 실시간 제어 시스템을 손쉽게 구현하는 것이 가능하도록 하였다. 개발된 DMC 블럭의 적용 가능성을 살펴보기 위하여 연구실에서 개발된 RCP 환경을 이용하여 DC 모터의 속도 제어 시스템을 구현하고 실시간 제어 실험을 수행하였다. 실험을 통해 비교적 빠른 시스템에도 DMC를 적용할 수 있음을 살펴볼 수 있었고 고전적인 PI 제어와의 차별성도 살펴보았다. 비교적 복잡한 제어기법인 DMC를 블럭화 함으로써 RCP 환경을 통해 손쉽게 DMC 기반의 제어시스템을 구현해 낼 수 있도록 만들었다는 것이 본 논문의 의의라고 할 수 있겠다.

REFERENCES

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

[2] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, “Generalized predictive control-Part I. The basic algorithm,” *Automatic*, vol. 23, no. 2, pp. 137-148, 1987.

[3] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, “Generalized predictive control-Part II. Extensions and interpretations,” *Automatic*, Vol 23, no. 2, pp. 149-160, 1987.

[4] C. E. Garcia and A. M. Morshedi, “Quadratic programming solution of dynamic matrix control (QDMC),” *Chem. Eng. Commun.*, vol. 46, pp. 73-87, 1986.

[5] Y. S. Lee, J. H. Kim, W. S. Kim, and O. K. Kwon, “Development of a rapid control prototyping system based on Matlab and USB DAQ boards,” *Journal of Institute of Control, Robotics, and Systems (in Korean)*,

vol. 18, no. 10, pp. 912-920, 2012.

[6] The Mathworks Inc., *Real-time workshop user's guide (rtw_ug.pdf)*, Version 6, 2005.

[7] The Mathworks Inc., *Real-time workshop embedded code user's guide (ecoder_ug.pdf),* Version 4, 2005.

[8] S. Rebeschie β, "MICROS-Microcontroller-based real time control system toolbox for use with Matlab/Simulink," *Proc. of IEEE Int. Symp. Computer Aided Control System Design*, pp. 267-272, 1999.

[9] K. H. Hong, W. S. Gan, Y. K. Chong, K. K. Chew, C. M. Lee, and T. Y. Koh, "An integarated environment for rapid prototyping of DSP algorithms using and Texas Instruments' TMS320C30," *Microprocessors and Microsystems*, vol. 24, no. 7, pp. 349-363, 2000.

[10] W. Lee, M. Shin, and M. Sunwoo, "Target-identical rapid control prototyping platform for model-based engine control," *Proc. Instrn Mech. Engrs Part D, J. Automobile Engineering*, vol. 218, pp. 755-765, 2004.

[11] D. Hercog and K. Jezernik, "Rapid control prototyping using Matlab/Simulink and a DSP-based motor controller," *Int. J. Engng ED.*, vol. 21, no. 3, pp. 1-9, 2005.

[12] R. Bucher and S. Balemi, "Rapid controller prototyping using Matlab/Simulink and Linux," *Control Engineering Practice*, vol. 14, pp. 185-192, 2006.

[13] C. F. Lin, C. Y. Tseng, and T. W. Tseng, "A hardware-in-the-loop dynamics simulator for motorcycle rapid controller prototyping," *Control Engineering Practice*, vol. 14, pp. 1467-1476, 2006.

[14] R. Kennel, "Improved direct torque control for induction motor drives with rapid prototyping system," *Energy Conversion and Management*, vol. 47, pp. 1999-2010, 2006.

[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," *Journal of Institute of Control, Robotics, and Systems (in Korean)*, vol. 20, no. 9, pp. 908-913, 2014.

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

[17] 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.

[18] K. J. Åström and L. Rundqwist, "Integrator windup and how to avoid it," *Proc. of the 1989 American Control Conference*, pp. 1693-1698, 1989.

[19] T. Meta, G, Y, Gyeong, J. H. Park, and Y. S. Lee, "Swing-up control of an inverted pendulum subject to imput/output constraints," *Journal of Institute of Control, Robotics, and Systems (in Korean)*, vol. 20, no. 8, pp. 835-84, 2014.

이 영 삼

제어 · 로봇 · 시스템학회 논문지, 제15권 제4호 참조.

유 광 명

2006년 부산대학교 전자전기통신공학부 졸업. 2008년 한국과학기술원 전기 및 전자공학과 졸업(석사). 현재 한전 전력연구원 선임연구원. 관심분야는 발전용 제어시스템 설계, 제어성능 최적화.