공학석사 학위논문

반복제어계를 이용한 무궤도 이동랙 동기화제어계 설계

Design of Synchronize Controller for Non-Rail Mobile Rack by using Repetitive Control Method

지도교수 하 윤 수

2016년 2월

한국해양대학교 대학원
제 어 계 측 공 학 과
박 진

본 논문을 박 진의 공학석사 학위논문으로 인준함

위원장 공학박사 진 강 규 (인)

위 원 공학박사 김 환 성 (인)

위 원 공학박사 하 윤 수 (인)

2016년 1월 15일

한국해양대학교 대 학 원

목 차

Design of Synchronize Controller for Non-Rail Mobile Rack by using Repetitive Control Method

Jin Park

Department of Control and Instrumentation Engineering,

Graduate School, Korea Maritime and Ocean University

Abstract

Non-rail mobile rack which is used as a cargo storage in logistics center can improve the storage capacities. Furthermore, It has an advantage to apply in traditional logistic center without change any renovation such as installing rail.

However when it is operated by a separated drive actuators which is mounted on the left and the right wheels, a precise position control for the wheels would be necessary even if unbalanced cargo weight on the non-rail mobile rack would be effected to the control.

Therefore, in this study, there is necessary to internal synchronize control for position tracking between left and right wheels on the non-rail mobile rack. Also an external synchronize control for same

straight movements between mobiles would be necessary.

For developing the internal and the external synchronize control algorithm, we suggest a synchronize control algorithm based on the repetitive control theory. Specially, an internal synchronize control algorithm with repetitive control theory use the robust servo control method due to parameter variations. In this case, we can set up the gains for robust servo control system by considering the cargo variations on the mobile rack. Also, for constructing the external synchronize control algorithm, we use a double repetitive control system to perform synchronize control between each mobile rack.

Finally, the efficiency of proposed control algorithm will be verified by simulation and experimental results. And the proposed algorithm would be applied to industrial area easily.

제 1 장 서 론

1.1 연구 배경과 연구방법

정보화·과학화 사회로 급격한 변화가 이루어지는 21세기는 “디지털 혁명”시대라 할 만큼 IT (Information Technology)기술이 급속히 발전을 하고 있다. 물류 산업에서도 정보기술을 기반으로 하는 무인자동화시스템의 정착이 빠르게 진행되고 있으며 이를 통해 물류비용을 최소화 하여 보다 싸게, 좀 더 빠르게 서비스하기 위해 물류관리체계가 발전되고 있지만 대부분의 첨단장비 및 물류 설비는 고가의 수입제품에 의존하고 있다.

무인자동화시스템의 정착이 빠르게 진행됨으로 인해 물류비용을 최소화하여 값싸고 빠른 시간 안에 소비자들에게 서비스하기 위해 물류관리체계가 혁신적인 발전을 하고 있는데 인터넷 쇼핑 및 홈쇼핑의 발달로 다품종 소량 주문에 신속함과 정확도를 요구하는 것이 늘어나고 있으며 의약품 등 고부가가치 물류 서비스에 대한 수요도 함께 증가하고 있다.

국내물류현장에서의 수입제품은 근무인력의 배치나 작업정서, 제품 및 물류 흐름 등이 선진국과는 다른 경우가 많아 국내 현장과 이질적인 경우가 많다. 따라서 우리나라 물류 환경에 부합하는 작업자 중심의 물류자동화의 실현을 통해 물류 업무 환경 개선으로 “근로자의 질” 향상에 기여하고 물류시간, 생산성 및 효율성을 증대시키기 위해 연구한다.

최근 화물이 다양해지고 입출고가 잦아짐에 따라 고속화, 유연화, 경량화 등을 고려한 물류작업 속도 향상 기술 실현의 필요성이 증대되고 있다. 제품을 보관하는 기능 외에 다양한 제품의 저장, 분류 및 입출고의 정확성과 신속성까지 겸비한 종합시설로의 발전이 필요하다. 이러한 시설을 통해 보관창고에서 운영 인력을 40% 절감시키는 효과와 보관효율 향상으로 매출액이 20% 증가할 수 있다.

따라서 최근 고정식 선반으로 인해 공간효율이 낮아지는 단점이 최근 부각되

고 있고 일본, 유럽 및 국내 등에서는 일부 관련 제품을 출시하고 있지만 물류 창고에서 활용 사례는 아직 많지 않다. 그리고 현재 국내 전동식 이동랙은 단순하게 모터에 의한 좌우 이동 및 안전장치 부탁으로 한정된 경우가 많다.

이번 연구에서는 무궤도 이동랙의 일반적인 형태, 구조 및 상태를 분석하고 이를 통하여 무궤도 이동랙에 대한 수학적 모델링을 통해 수치적인 시스템 분석과 해당 시스템이 가지고 있는 특성을 확인한다. 이러한 시스템의 상태를 통해 적절한 제어기를 선정하여 알맞게 설계하고 이를 검증하기 위한 시뮬레이션을 진행한다.

또한 이번 연구에서는 단일 형태로 되어 각각의 동작만을 수행하는 이동랙뿐만 아니라 실제 물류 현장의 작업 환경 및 장비 현황을 고려하여 다중 이동 모듈에 관한 알고리즘도 함께 고려하여 연구를 진행한다.

하지만 한가지 고려해야할 사항은 실제 물류현장의 구조 및 환경은 창고의 전체적인 형태, 주로 보관되어져 있는 화물의 분류 상태 그리고 화물의 입·출하 작업 방식에 따라 rack의 크기와 배열 방법이 달라질 수 있어서 이러한 모든 상태를 고려한 연구는 현실적으로 어려울 수 있으니 일반적인 프레임을 고려하여 연구를 진행한다.

이번 이동랙의 제어 기술 연구의 목적은 무궤도 이동랙의 문제점에 의해 고려되었다. 이동랙의 주행 궤적이 조금 어긋나는 것이 크게 문제가 되지 않을 것 같지만 이동랙에 화물 작업을 진행하기 위해 이동을 지속적으로 진행하게 된다.

이때 아무리 작은 오차가 발생하더라도 긴 거리를 움직이거나 이 오차가 누적이 되기 시작하면 정해진 이동 위치에서 많이 벗어나게 된다. 그리고 현재 개발 중인 이동랙의 경우 사용자의 필요에 따라 다련식으로 나눈 다중 이동 모듈로 단일 열로 이루어 진 랙이 아니라 여러 개의 이동랙이 한 열을 이루고 있다.

따라서 이동 궤적이 벗어나게 되면 이동랙 간의 충돌이 발생하게 되며 이렇게 되면 장비와 적재 화물의 훼손 그리고 더 나아가 작업자가 다치게 되는 인명 피해까지 발생할 수 있다. 따라서 이동랙의 좌우 구동부의 이동을 일정하게

제어하여 이동랙의 위치가 주행 방향과 일치하도록 조정하기 위해 안정적인 이동을 연구하고 구현한다.

1.2 논문의 구성

본 연구의 구성은 다음과 같다. 제2장에서 일반적인 이동랙에 대하여 간략하게 설명하며, 일반적인 물류창고에서 이용하고 있는 고정형 랙과 이동형 랙과의 차이점에 대해 알아본다. 그리고 이동랙의 형태에서 궤도형과 무궤도형 이동력의 특징에 대해서도 고찰하도록 한다.

이동랙의 수학적인 모델링을 위하여, 팔레트 3단 적재형을 가정하고 화물의 무게, 이동랙 베이스 무게 및 관련사항을 고려하여 선형미분방정식 및 상태방정식을 도출하도록 한다. 이때, 랙 구조물의 강성변화를 고려하여 탄성변화율의 변동성을 검토하며, 이러한 변화를 시스템에 작용하는 외란으로 가정하여 모델링을 행한다.

제3장에서는 이동랙에 대한 전반적인 동기화 제어기 설계를 구성한다. 이때 각각의 구동부에서는 강인 서보 제어계를 통해 적절하게 구동부 동작을 제어하도록 한다. 또한 무궤도 이동랙의 경우 주행 보정을 위해 동기화 제어를 구성하도록 하며, 내부 동기화 제어를 통하여 좌측과 우측 구동부의 출력을 조정하여 이동랙내의 동기화 제어를 행한다. 한편, 외부 동기화 제어에서는 각 이동랙간의 동기화 제어를 수행하며, 이동랙이 이동시 발생하는 오차에 대하여 반복적으로 게인을 수정하여 제어를 행하는 반복제어기법을 활용하도록 한다. 또한, 상기의 내부 및 외부 동기화 제어계의 시뮬레이션을 통하여 그 유효성을 확인한다.

제4장에서는 시뮬레이션을 통해 확인한 제어계를 실제 실험 모델이 적용하도록 한다. 이용된 모형은 약 1/8 크기의 미니모델을 이용하며, 제어부는 Arduino Due를 이용하였으며 구동부는 소형 기어드 DC 모터로 구성하였다. 이동랙의 주행거리는 DC 모터에 부착된 엔코더를 이용하여 측정하였으며 이를 수집하여 실험 결과를 분석하도록 한다.

제 2 장 이동랙 시스템

본 장에서는 이동랙에 대해서 대략적인 기능과 함께 각 이동랙 종류에 대한 특징들을 정리하도록 한다. 또한 수학적인 모델링을 위해서는 이동랙을 기능별로 정리하며, 각 환경 변화를 고려하고 시뮬레이션을 통하여 변동성을 검토하도록 한다.

그림 2.1 이동랙의 기본 형태

Fig. 2.1 Basic form of mobile rack

2.1 이동랙 시스템

2.1.1 이동랙의 특징

기존의 고정랙은 그림 2.2을 보면 알 수 있듯이 지게차나 작업자가 적재장소로 이동하기 위한 주행로와 각 고정랙간의 작업 동선으로 구성되며, 이는 작업이 이루어지지 않을 때에는 공간 손실로 발생되는 부분이다. 즉 이러한 공간들은 항상 비워져 있으므로 물류센터 공간 활용상 효율적이지 못한 것으로 공간의 활용도가 많이 떨어진다고 볼 수 있다.

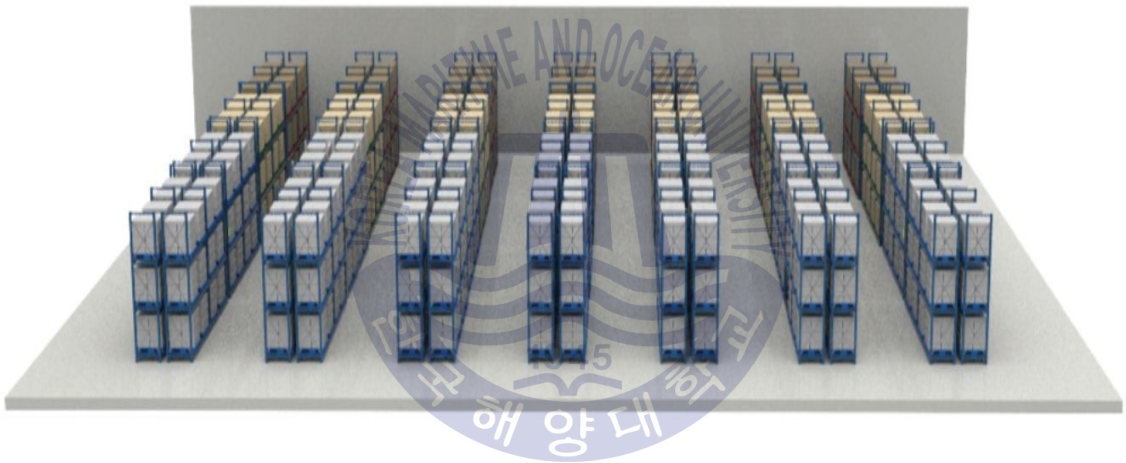

그림 2.2 일반적으로 설치된 랙
Fig. 2.2 Installed rack in common

이동랙은 고정랙에 비해 보관 효율이 약 75%까지 증가하게 되며, 이러한 이동랙의 경우 사용되는 목적이나 설치되는 환경 특성에 따라 고층화 할 수도 있으므로 물류 창고의 공간 활용도를 더욱 더 높게 할 수 있다. 그림 2.2와 그림 2.3을 비교해보면 알 수 있듯이 고정랙 사이의 작업통로를 대폭 줄일 수 있으므로, 이를 이동식 랙의 보관 공간으로 모두 사용할 수 있다.

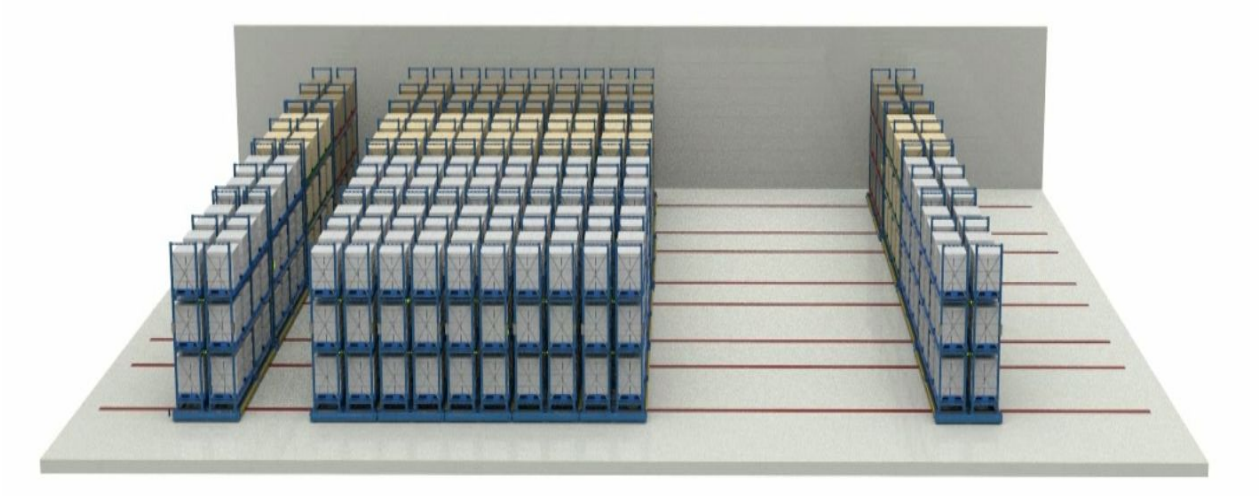

그림 2.3 일반적인 이동랙
Fig. 2.3 Mobile rack in common

2.1.2 이동랙의 종류

일반적으로 이동랙이란 화물을 적재하는 랙을 이동이 가능하도록 제작하여 효율적인 공간 활용을 구현하기 위한 것으로 양쪽 끝은 일반적으로 고정랙을 설치하며, 그 사이에는 이동랙을 위치시키며 필요한 작업 위치에만 작업 통로를 만들어 개폐하고 해당 위치의 작업을 수행한다.

그림 2.4 궤도형 이동랙의 예시
Fig. 2.4 Example of rail type mobile rack

이러한 이동랙에는 크게 두 가지 종류의 이동랙으로 분류되는데 궤도형 이동랙과 무궤도형 이동랙으로 나뉜다. 궤도형 이동랙은 바닥에 궤도가 설치되어 있는 형태로 해당 궤도를 따라 구동휠이 움직이므로 이동 궤적에 따라 움직여서 직성 주행에는 안정적이지만 궤도를 사전에 설치해야 하는 단점이 존재한다.

무궤도형식의 이동랙은 궤도가 없는 형태로서, 이동랙의 직진성을 확보하기 위하여 양쪽에 위치한 구동휠을 정밀하게 제어할 필요가 있다. 이러한 형태는 물류센터 내부에 레일 설치 작업 등의 추가 작업이 필요 없으며 이동랙 베이스 모듈만 설치가 되면 작동이 가능하다.

이동랙 베이스 모듈의 기본항목으로서는 비틀림방지 X자 브레이싱, 독립구동형 모터 결속부, 구동모듈 차폐부, 미끌림 방지 구동휠, 접촉식 자동 충전장치, 주행 궤적센서, 이동랙 안전센서로 이루어져 있다. 이동랙 베이스 모듈은 바닥면에 위치해 있고, 그 위로 이동랙 프레임을 설치하는 형식이다. 베이스모듈에는 이동랙을 이동시키기 위한 구동부 및 동력 전달 부분이 있고 궤도 위를 따라 움직일 수 있도록 바퀴가 설치되어 있다. 이것은 국내에 대부분 설치되어 사용되고 있는 이동랙의 형태이다.

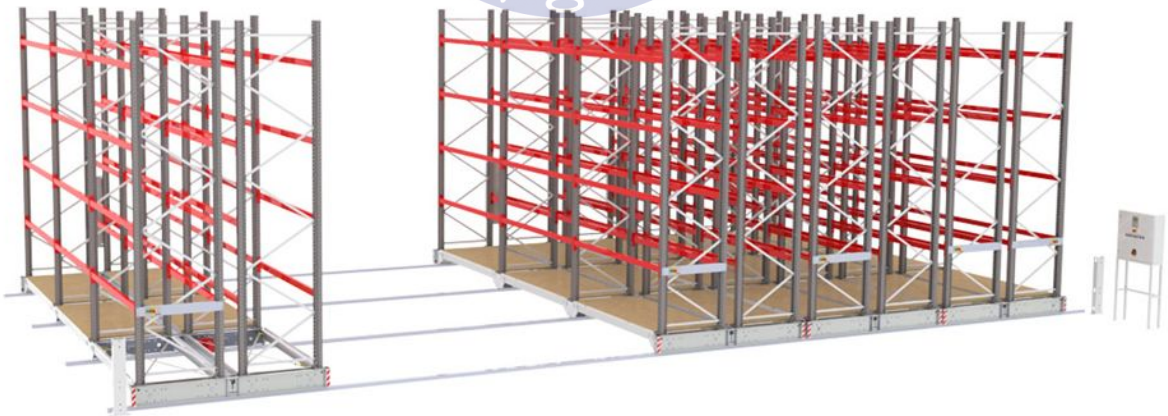

그림 2.6 구성부로만 표현한 이동랙

Fig. 2.6 Mobile rack which was expressed only a configuration part

위에서 언급한 궤도형 이동랙의 장점은 주행 방향으로 구동시에 추가적인 제어가 필요 없으며 전진 및 후진 동작을 할 때에 구동 및 정지에 따른 이동 명령만 적절하게 이루어지면 어느 정도의 작동이 가능하다.

하지만 단점으로는 이를 설치할 때에 추가 공사가 필요하며 설치되어 있는 궤도의 위치가 작업자들의 이동경로와 일치하므로 지게차 혹은 사람이 이동할 때 궤도로 인하여 걸림 등이 발생하기도 한다.

일반적인 궤도형 이동랙의 경우 하나의 모터를 이용해 구동부를 구성할 수 있지만 무궤도 이동랙의 경우 주행 궤적을 일정하게 잡아주는 부분이 없으므로 최소 양쪽 두 곳 이상의 모터를 이용해 이동 명령을 수행한다. 그렇게 함으로서 구동 중 이동랙이 방향이 틀어지는 현상이 발생해도 좌우 모터의 토크 조절을 통해 원래대로 작동할 수 있기 때문이다.

2.2 이동랙 모델링

본 연구에 고려된 이동랙은 3단으로 구성된 형태로서, 구동부 및 동력 전달부는 이동랙 베이스에 위치하고 있으며, 상부의 랙에 비하여 구성요소의 증가로서 무게가 더 증가한다. 그리고 각 단에서 화물이 적재되고 이동랙이 구동될 때에는 각 화물 및 이동랙 자체 무게에 의한 특성을 고려해야 한다.

본 이동랙 모델링은 실제로 사용되고 있는 이동랙을 기준으로 설계되어진 무궤도 이동랙의 구조 방식이며, 설계를 통하여 일반적으로 사용되어지는 프레임 재질을 통해 탄성계수를 계산하고 이를 적용하여 전체적인 모델링을 하였다.

앞서 설명한 이동랙의 대략적인 형태 및 구성에 따른 다이어그램과 이를 이용한 모델링은 그림 2.4와 같이 나타난다. 좌측은 이동랙의 도면을 간략하게 표현한 그림이며 우측은 이를 토대로 그려낸 모델링의 개요이다. 각 베이스 모듈 및 각 층의 랙 무게 그리고 각 층에 적재된 화물의 무게를 고려하며 높이와 이동랙의 재질에 따른 탄성계수를 고려하였다.

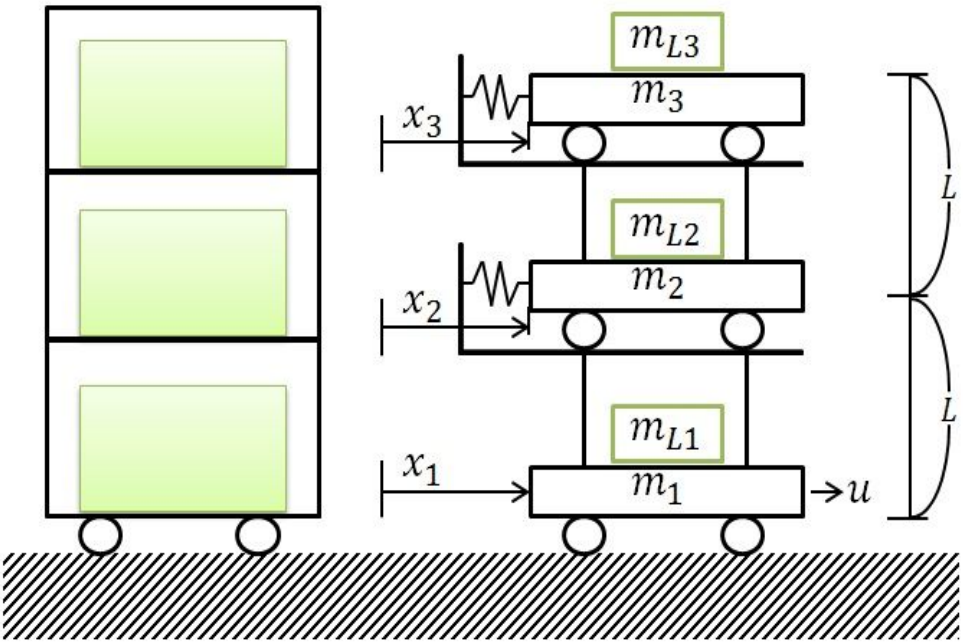

그림 2.7 이동랙의 간략화된 2D 도면과 모델링

Fig. 2.7 Simplified 2D drawing and modeling of the moving rack

$$m_{1L3}\ddot{x}_1 + k(x_1 - x_2) + B\dot{x}_1 = u \tag{1a}$$

$$m_{2L3}\ddot{x}_2 + k(x_2 - x_1) + k(x_2 - x_3) = 0 \tag{1b}$$

$$m_{3L3}\ddot{x}_3 + k(x_3 - x_2) = 0 \tag{1c}$$

여기서, $m_{1L3} = m_1 + m_{L1} + m_2 + m_{L2} + m_3 + m_{L3}$, $m_{2L3} = m_2 + m_{L2} + m_3 + m_{L3}$, $m_{3L3} = m_3 + m_{L3}$, $k = 2(\frac{3EI}{L^3})$이며, E 는 수직구조물 재료의 탄성계수이며 I는 단면 2차 모멘트를 나타낸다. 이때, $I = \frac{d_2 d_1^3 - (d_2 - d_t)(d_1 - d_t)^3}{12}$ 이고 $d_1 = d_2$ 은 단면길이를 나타내며 d_t 는 두께를 나타낸다.

상기에서 이동랙 수직구조물에 비하여 화물의 중량이 상대적으로 무거운 것을 고려한다면, 이동랙의 주요 움직임에 가장 큰 영향을 미친다고 볼 수 있다. 위의 운동방정식을 고려하여 상태방정식을 세우면 다음과 같다.

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2a}$$

$$y(t) = Cx(t) \tag{2b}$$

여기서,

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{-k}{m_{1L3}} & \frac{k}{m_{1L3}} & 0 & \frac{-B}{m_{1L3}} & 0 & 0 \\ \frac{k}{m_{2L3}} & \frac{-2k}{m_{2L3}} & \frac{k}{m_{2L3}} & 0 & 0 & 0 \\ 0 & \frac{k}{m_{2L3}} & \frac{-k}{m_{3L3}} & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{m_{1L3}} \\ 0 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix}$$

위의 상태방정식에 대해서 이동랙의 파라미터를 표 2.1과 같이 고려한다.

Symbol	Value	Unit	Symbol	Value	Unit
m_1	50	$[kg]$	E	2.05×10^{11}	$[kgf/cm^2]$
m_2	25	$[kg]$	d_1	85	$[mm]$
m_3	25	$[kg]$	d_2	55	$[mm]$
m_L	6000	$[kg]$	d_{t1}	3.2	$[mm]$
L	1.3	$[m]$	d_{t2}	2.3	$[mm]$

표 2.1 이동랙의 파라미터

Table 2.1 Parameter of mobile rack

2.3 이동랙 환경변화 특성

이동랙 모델링에서 랙 구조물의 탄성변화율을 $k = k_n + \Delta k$ 로 표현하여 상태 방정식을 다음과 같이 나타낼 수 있다. 여기서, k_n은 공칭 탄성계수를 나타내고 Δk는 탄성계수의 변화율을 나타낸다.

$$\begin{bmatrix} \dot{x}_s \\ \ddot{x}_s \end{bmatrix} = \begin{bmatrix} 0 & I \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_s \\ \dot{x}_s \end{bmatrix} + Bu \tag{3a}$$

$$= \begin{bmatrix} 0 & I \\ A_{n21} & A_{22} \end{bmatrix} \begin{bmatrix} x_s \\ \dot{x}_s \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta A_{21} \end{bmatrix} x_s + B_u \tag{3b}$$

여기서 $x_s = [x_1 x_2 x_3]^T$ 이며,

$$A_{n21} = \begin{bmatrix} \frac{-k_n}{m_{1L3}} & \frac{k_n}{m_{1L3}} & 0 \\ \frac{k_n}{m_{2L3}} & \frac{-2k_n}{m_{2L3}} & \frac{k_n}{m_{2L3}} \\ 0 & \frac{k_n}{m_{2L3}} & \frac{-k_n}{m_{3L3}} \end{bmatrix}, \Delta A_{n21} = \begin{bmatrix} \frac{-\Delta k}{m_{1L3}} & \frac{\Delta k}{m_{1L3}} & 0 \\ \frac{\Delta k}{m_{2L3}} & \frac{-2\Delta k}{m_{2L3}} & \frac{\Delta k}{m_{2L3}} \\ 0 & \frac{\Delta k}{m_{2L3}} & \frac{-\Delta k}{m_{3L3}} \end{bmatrix}$$

이다.

또한, k는 여러 변수에 의해 변화하지만 여기서는 랙의 재질 두께 변화에 따른 단면 2차 모멘트를 이용해 탄성계수의 변화량을 구하며, 다음과 같이 나타낸다.

여기서 d_t는 두께를 나타내며 Δd_t는 두께의 변화량을 나타낸다.

$$k = 2\frac{(3EI)}{L^3} = 2\left(\frac{3EI + 3E\Delta I}{L^3}\right) = k_{n1} + \Delta k_1 \tag{4}$$

$$\Delta I = I - \frac{(d_2 + \Delta d_t)(d_1 + \Delta d_t)^3}{12} - \frac{(d_2 - d_t + \Delta dt)(d_1 - d_t + \Delta d_t)^3}{12} \tag{5}$$

상기 이동랙의 두께 변화율에 대한 탄성계수변화 그래프는 다음과 같다.

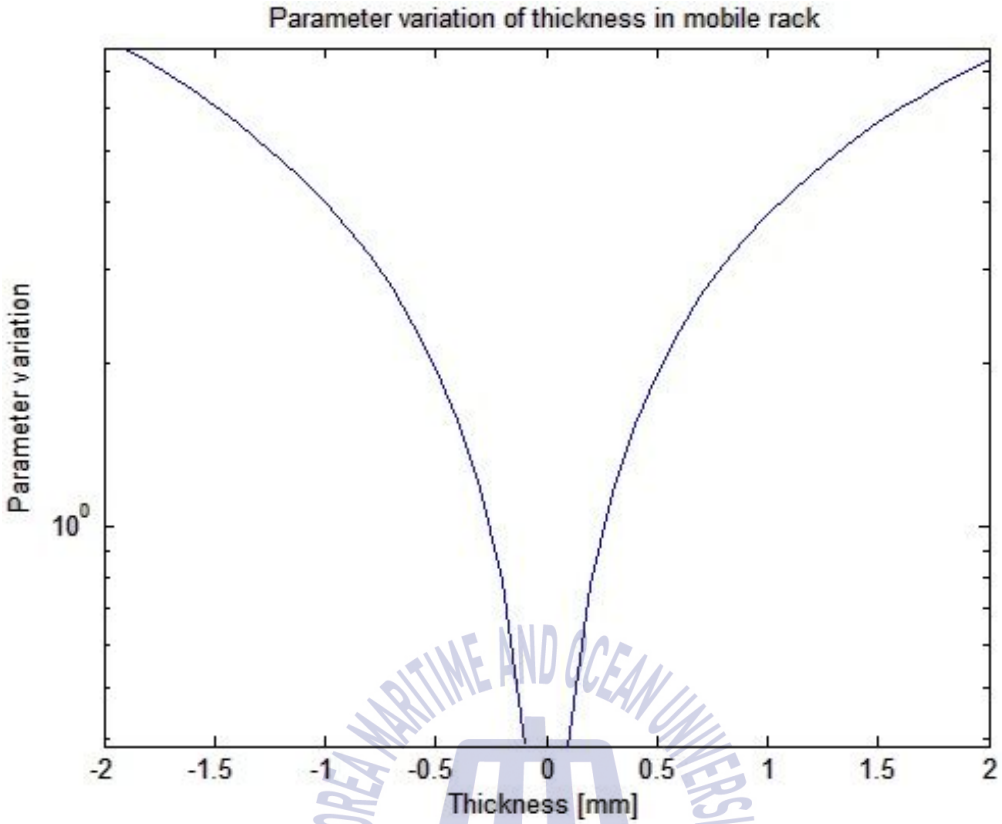

그림 2.8 이동랙 두께의 파라미터 변화량

Fig. 2.8 Parameter variation of thickness in mobile rack

그래프를 보면 두께의 변화에 따라 파라미터가 변화하는 것을 확인할 수 있다. 이와 같이 변화하는 파라메터 변동값을 시스템의 외부에서 인가되는 외란으로 가정할 수 있고 이를 이용하여 파라메터 변화에 강인한 서보제어기를 설계하고자한다.

이에 대한 설명과 제어기에 관한 자세한 내용은 제 3장에서 연구하고 시뮬레이션하여 확인해본다.

제 3 장 동기화 제어기 설계

3.1 강인 서보제어기

본 연구에서는 파라미터 변동에 대해서는 강건한 강인 서보제어기를 구성하여 이용하며, 작업 상황에 따라 화물분포가 수시로 변하고 또 그 변동의 폭이 큰 이동랙의 특징으로 인해 다양한 화물변동구간에서 이동랙의 움직임을 유효하게 제어할 수 있을 것으로 고려된다.

상기 서보 제어계는 각 구동부를 동작할 때에 각각 구동부에 적용하여 좌우측 구동부의 동기화 제어 혹은 다중 이동랙 별로 동기화 제어를 진행할 때에 더욱 안정적이고 효율적으로 제어하도록 한다.

3.1.1 강인 서보제어기의 구성

제2장에서 논의한 바와 같이 탄성계수의 변동을 시스템의 외부에서 인가되는 외란으로서 가정할 수 있으므로 로바스트 서보계설계법에 의해 제어계를 구성할 수 있다.

로바스트 서보계 설계법은 그 루프내에 목표 입력이나 외란 입력의 발생기구에 대한 모델(내부모델)을 포함하고 있어야 한다. 이로서, 서보계는 다음 그림과 같은 구성으로 된다.

일반적으로 파라미터 고정일 경우와 부분적 파라미터 변동을 고려할 경우에는 제어대상의 내부모델을 위하여 그림 3.1과 같이 구성하며, 제어대상의 모든 파라미터 변동을 고려할 경우에는 그림 3.2와 같이 구성한다.

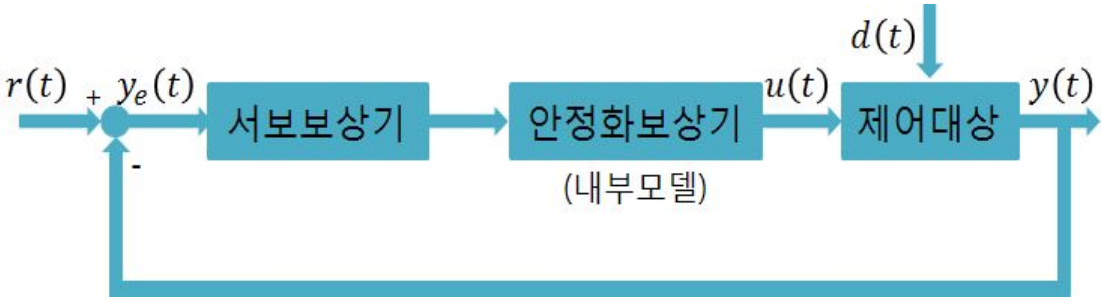

그림 3.1 부분적 파라미터 변동을 고려한 서보시스템

Fig. 3.1 Servo System with a partial parameter variations

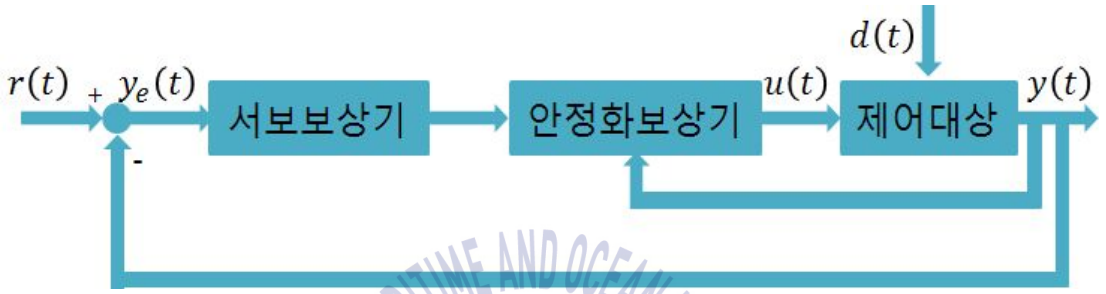

그림 3.2 모든 파라미터 변동을 고려한 서보시스템

Fig. 3.2 Servo System Considering all of the parameter variations

또한, 본 연구에서 설계할 서보계의 설계법은 상태피드백을 관측기에 의해 실현될 수 있고, 아래와 같은 제어대상의 경우에는 그림 3.2와 같이 구성할 수 있다.

$$\dot{x}(t) = Ax(t) + Bu(t) + d(t); x(0) = x_o \tag{6a}$$

$$y(t) = Cx(t) \tag{6b}$$

최적 로바스트서보계의 설계법에서는 $t \to \infty$에서 $y(t)$가 목표치 $r(\neq 0)$에 일치한다고 가정하면, 입력 $u(t)$도 $t \to \infty$에서 '0'으로 되지 않는다. 따라서 다음 식의 2차 형식 평가함수를 이용하면 그 값은 무한대로 되어, 식 (7)과 같은 평가함수를 이용할 수 없다.

$$J = \int_0^\infty (\| x(t) \|_Q^2 + \| u(t) \|_R^2) dt \tag{7}$$

이때 $u(t)$가 $t\to\infty$에서 일정치를 가지므로 $\lim_{t\to\infty}\dot{u}(t)=0$으로 된다. 따라서, $v(t)=\dot{u}(t)$를 제어 입력으로 하여 식 (6)을 다시 정리하면 다음과 같이 된다.

$$\dot{x}_e(t)=A_e x_e(t)+B_e v(t)+D_e d(t) \tag{8a}$$

$$y(t)=C_e x_e(t) \tag{8b}$$

단, $x_e(t)=[x(t)^T u(t)^T]^T$ 이고

$$A_e=\begin{bmatrix}A & B\\ 0 & 0\end{bmatrix}, B_e=\begin{bmatrix}0\\ I_m\end{bmatrix}, D_e=\begin{bmatrix}I_n\\ 0\end{bmatrix}, C_e=[C\ 0] \tag{9}$$

이다.

한편, $t\to\infty$에서 $x(t)$와 $u(t)$가 일정치 x_s, u_s를 취한다고 하면, 이 값은 식 (8)의 정상해

$$\begin{bmatrix}0\\ r\end{bmatrix}=\begin{bmatrix}A & B\\ C & 0\end{bmatrix}\begin{bmatrix}x_s\\ u_s\end{bmatrix}+\begin{bmatrix}d\\ 0\end{bmatrix} \tag{10}$$

로부터, 다음과 같이 구해진다.

$$\begin{bmatrix}x_s\\ u_s\end{bmatrix}=\begin{bmatrix}A & B\\ C & 0\end{bmatrix}^{-1}\begin{bmatrix}-d\\ r\end{bmatrix}=Z^{-1}\begin{bmatrix}-d\\ r\end{bmatrix} \tag{11}$$

여기서, 정상치로부터의 변동분 $\delta x(t)$, $\delta u(t)$

$$\delta x(t)=x(t)-x_s \tag{12a}$$

$$\delta u(t)=u(t)-u_s \tag{12b}$$

를 고려하여, $\delta x_e(t)=[\delta x(t)^T \delta u(t)^T]^T$라 두면 식 (8)은

$$\delta \dot{x}_e(t) = A_e \delta x_e(t) + B_e v(t) \tag{13a}$$

$$y(t) - r = C_e \delta x_e(t) \tag{13b}$$

와 같이 고쳐 써진다. 따라서, 이 시스템에 있어서, $\delta x_e(t) \rightarrow 0$으로 하는 제어를 행하면, $y(t) \rightarrow r$이 달성 되는 것을 알 수 있다.

따라서, $\delta x_e(t) \rightarrow 0$으로 하는 제어는 최적레귤레이터를 이용해서 실현할 수 있다. 즉, 평가함수를

$$J_e = \int_0^\infty (\| x_e(t) \|_{Q_e}^2 + \| u(t) \|_{R_e}^2) dt \tag{14}$$

단, Q_e는 $(n+m)\times(n+m)$의 반정정대칭행렬, R_e는 $m \times m$ 대칭행렬로 정의하여, 이것을 최소로하는 최적레귤레이터 문제를 풀면 된다. 여기서, $Q_e = C_e^T C_e$라 하면, 평가함수 식 (14)는

$$J_e = \int_0^\infty (\| y(t) - r \|^2 + \| c(t) \|_{R_e}^2) dt \tag{15}$$

로 정리되며, 편차와 제어입력의 미분에 하중이 가해진 평가함수로 된다.

식 (14)를 최소로 하는 최적레귤레이터의 해는

$$v(t) = - F_e \delta x_e(t) \tag{16}$$

단,

$$F_e = R_e^{-1} B_e^T P_e \tag{17}$$

로 주어진다. 여기서, $(n+m)\times(n+m)$ 행렬 P_e는 리카티방정식

$$A_e^T P_e + P_e A_e + Q_e - P_e B_e R_e^{-1} B_e^T P_e = 0 \tag{18}$$

의 정정대칭해이다.

식 (16)은 $\delta x(t)$와 $\delta x(t)$의 피드백으로 나타내어지나, 제어계 설계를 위해서는 변경할 필요가 있다. 즉, 상태 $x(t)$와 편차 $y_e(t)=r-y(t)$의 피드백형태로 변환시켜 식 (16)에 식 (12)을 대입하면, 다음과 같은 형식으로 표현된다.

$$v(t)=-F_e\begin{bmatrix}x(t)\\u(t)\end{bmatrix}+F_e\begin{bmatrix}x_s\\u_s\end{bmatrix} \tag{19}$$

한편, 식 (6)에 의해

$$\begin{bmatrix}\dot{x}(t)\\y(t)\end{bmatrix}=Z\begin{bmatrix}x(t)\\u(t)\end{bmatrix}+\begin{bmatrix}d\\0\end{bmatrix} \tag{20}$$

이므로, 이 식과 식 (11)을 이용하면, 식 (19)는

$$v(t)=-F_eZ^{-1}\begin{bmatrix}\dot{x}(t)-d\\y(t)\end{bmatrix}+F_eZ^{-1}\begin{bmatrix}-d\\r\end{bmatrix}=-F_eZ^{-1}\begin{bmatrix}\dot{x}(t)\\y(t)-r\end{bmatrix} \tag{21}$$

과 같이 고쳐 써진다. 여기서

$$F_eZ^{-1}=[K_1\,K_2] \tag{22}$$

라 놓고, 식 (21)을 0으로부터 t까지 적분하면

$$u(t)=-K_1x(t)+K_2\int_0^t y_e(t)dt+K_1x(0) \tag{23}$$

로 된 제어칙이 얻어진다. 따라서, 초기상태 $x(0)=0$일 때, 그림3.3에 나타낸 제어계는 최적로바스트 서보계로 되고 상태방정식과 출력방정식은 식 (24)와 같다.

$$\frac{d}{dt}\begin{bmatrix}x(t)\\y(t)\end{bmatrix}=\begin{bmatrix}A-BK_1 & BK_2\\-C & 0\end{bmatrix}\begin{bmatrix}x(t)\\u(t)\end{bmatrix}+\begin{bmatrix}d(t)\\r(t)\end{bmatrix} \tag{24a}$$

$$y(t) = [C\ 0]\begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \tag{24b}$$

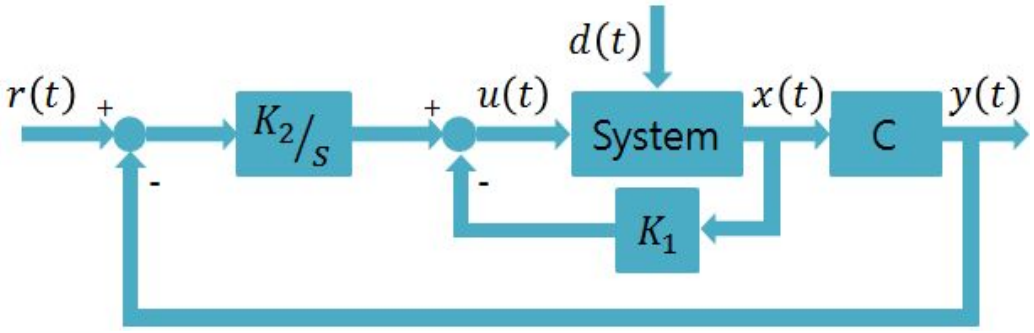

그림 3.3 제어계의 블록다이어그램

Fig. 3.3 Block diagram of control system

[정리 1][1)]

가제어성 및 가관성을 만족하는 m 입력 p 출력 선형시불변시스템 제어대상에 대하여 임의의 정수 l_i $(\geq 0; i = 1,\cdots,p)$에 대하여 $[l_1 \cdots\ l_p]$ 형 제어계가 구성 가능하기 위한 필요충분조건은 다음식을 만족하는 것이다

$$rank\begin{bmatrix} A & B \\ C & 0 \end{bmatrix} = n + p \tag{27}$$

3.1.2 시뮬레이션

위의 [정리1]을 만족하는 최적로바스트 서보계를 구성 조건을 확인하기 위하여 다음과 같이 rank 조건을 검토하면,

1) 古田勝久, 川路茂保, 美多勉, 原辰次, "메카니칼시스템 제어"

$$rank\,Z = rank\begin{bmatrix} A\,B \\ C\,0 \end{bmatrix} = rank\begin{bmatrix} 0 & 0 & 0 & 1 & 0\,0 & 0 \\ 0 & 0 & 0 & 0 & 1\,0 & 0 \\ 0 & 0 & 0 & 0 & 0\,1 & 0 \\ \dfrac{-k}{m_{1L3}} & \dfrac{k}{m_{1L3}} & 0 & \dfrac{-B}{m_{1L3}} & 0\,0 & \dfrac{1}{m_{1L3}} \\ \dfrac{k}{m_{2L3}} & \dfrac{-2k}{m_{2L3}} & \dfrac{k}{m_{2L3}} & 0 & 0\,0 & 0 \\ 0 & \dfrac{k}{m_{2L3}} & \dfrac{-k}{m_{3L3}} & 0 & 0\,0 & 0 \\ 1 & 0 & 0 & 0 & 0\,0 & 0 \\ 0 & 1 & 0 & 0 & 0\,0 & 0 \\ 0 & 0 & 1 & 0 & 0\,0 & 0 \end{bmatrix} = 7 \qquad (28)$$

로서, Full rank를 만족하고 있음을 알 수 있다.

이에 대하여 로바스트 서보계 게인을 구하기 위하여 다음과 같이 확대계를 구성하며,

$$A_e = \begin{bmatrix} A\,B \\ 0\;0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0\,0 & 0 \\ 0 & 0 & 0 & 0 & 1\,0 & 0 \\ 0 & 0 & 0 & 0 & 0\,1 & 0 \\ \dfrac{-k}{m_{1L3}} & \dfrac{k}{m_{1L3}} & 0 & \dfrac{-B}{m_{1L3}} & 0\,0 & \dfrac{1}{m_{1L3}} \\ \dfrac{k}{m_{2L3}} & \dfrac{-2k}{m_{2L3}} & \dfrac{k}{m_{2L3}} & 0 & 0\,0 & 0 \\ 0 & \dfrac{k}{m_{2L3}} & \dfrac{-k}{m_{3L3}} & 0 & 0\,0 & 0 \\ 0 & 0 & 0 & 0 & 0\,0 & 0 \end{bmatrix},\; B_e = \begin{bmatrix} 0 \\ I_m \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad (29)$$

또한, 하중행렬을 다음과 같이 두도록 하자.

$$Q_e = C_e^T C_e = [C\,0]^T[C\,0] = \begin{bmatrix} 1\,0\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0 \end{bmatrix},\; R_e = 1 \qquad (30)$$

위의 조건에 의해 최적 피드백 이득 F_e는 Matlab을 통해 구해지면 K_1, K_2를

계산해 내면 다음과 같이 얻을 수 있다.

$$K_1 = [59.0445\ 0.1108\ 0.0554\ 6.4153\ 3.7690\ 1.8850] \tag{31a}$$

$$K_2 = 1.7321 \tag{31b}$$

상기 주어진 로바스트 서보계 게인을 이용하여 제어계를 구성하고 시뮬레이션을 행한 제어 결과는 그림 3.4와 같이 나타난다. 결과에서 알 수 있듯이 고중량 화물로 고려된 상태로 인해 기본 스텝응답의 경우 약간의 오버슈트가 발생하는 것을 확인할 수 있다. 이러한 문제점은 적절한 하중계수조정에 이루어질 수 있으며, 또한 반복제어계 설계를 통해 구해질 수 있다.

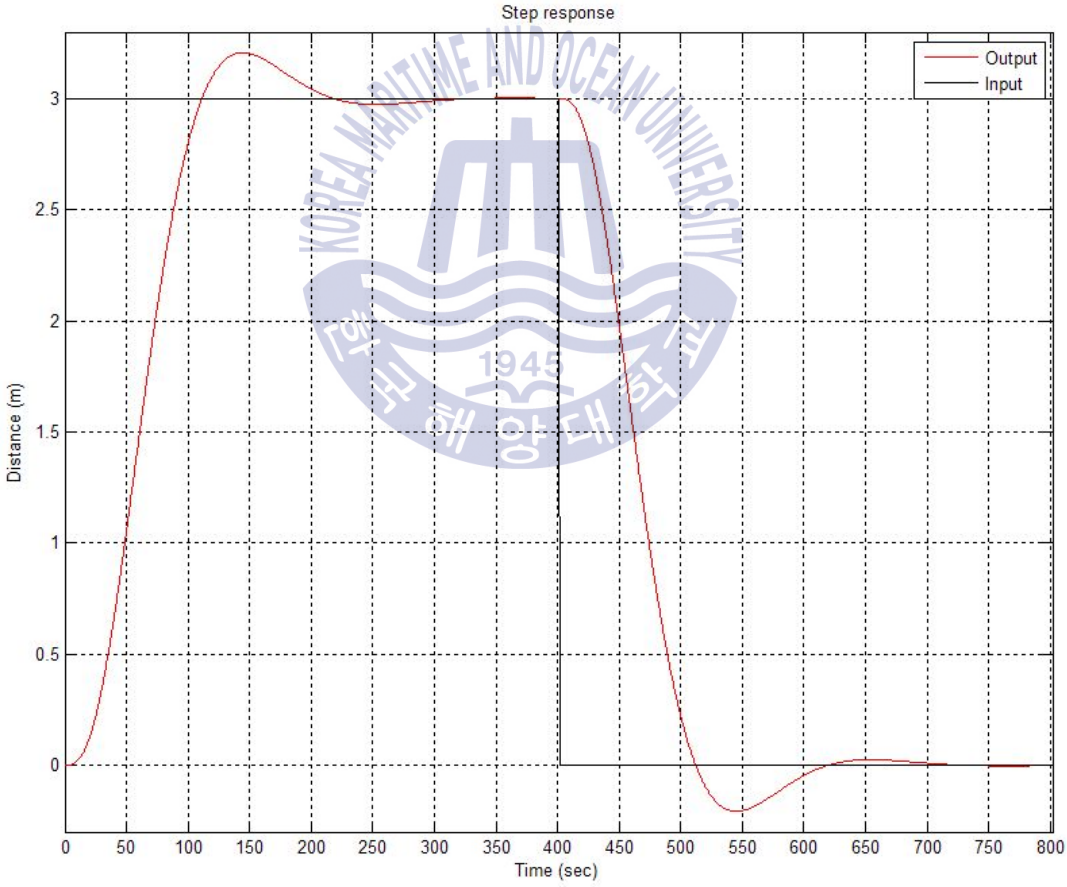

그림 3.4 서보제어기가 적용된 후 스텝 응답

Fig. 3.4 Step response on system with servo control

3.2 반복제어계

3.2.1 반복제어계 구성

위에서 구해진 이동랙 상태방정식과 설계진 제어계 파라미터를 이용하여 제어시스템을 구성한 결과, 강인 서보 제어계의 출력응답은 그림 3.4와 같이 나타날 수 있다. 상기 그래프에서 오버슈트가 발생하는 것을 알 수 있으며, 이러한 것은 이동랙의 화물하중에 의해 관성이 작용한 것으로 판단된다. 이에 대하여 로바스트 서보계 설계시 적절한 하중계수 선정이 필요하며, 또한 본 연구에서 추가하는 반복제어계 설계를 이용하였는데 이러한 응답은 이동랙에 고려된 무게가 영향이 있는 것 같다. 따라서 이를 조금 더 안정하게 제어하기 위해 다음과 같이 반복 제어기를 설계하여 적용한다.

먼저, 상기 이동랙은 앞서 언급한 내용과 같이 3가지 요소인 이동랙의 형태, 프레임의 재질 및 적재화물의 질량이 이동랙 구동에 큰 영향을 끼친다. 이는 실제 현장을 고려한 화물 하중으로서 각 단마다 6톤으로서 총 18톤에 이른다. 또한 이동랙의 높이가 높아서 이동랙 이동시에 무게중심의 영향을 많이 받는데 비하여 구동력이 가해지는 구동부는 최하층인 베이스 모듈에 장착이 되므로 매우 불안정하게 이동을 하게 된다.

이러한 요인으로 인하여, 이동랙의 파라미터 변화에 대하여 강건한 서보 제어계를 적용하여도 오버슈트가 발생하는 등의 다소 만족스럽지 못한 결과를 보여주고 있다. 즉, 각 단의 및 전체적으로 흔들림은 적으나 목표 도달시에 오버슈트가 발생하는 것을 볼 수 있다. 이러한 오버슈트는 이동랙간의 충돌을 야기시키게 되어 실제 구현시에는 크나 큰 문제를 야기시키게 된다.

따라서 목표치에 큰 흔들림이 없이 정확히 도달하는 것이 중요하다고 보이며 특정 위치에 정확하게 정지를 해야만 다른 이동랙 간의 충돌을 방지할 수 있으므로 오버슈트 문제는 반드시 해결되어야 한다. 이러한 문제로 인하여 본 연구에서는 반복제어기를 이용하여 이동랙의 이동을 제어하며, 작업상황에 따라 변

하는 이동랙의 하중 변화에 대하여 지속적으로 제어가 가능한 설계기법이 바람직할 것으로 보여진다.

3.2.2 반복제어계 설계

임의의 주기의 목표신호를 추종 목표로 하는 경우에 대하여 내부모델원리를 고려하면 일정한 주기를 가지는 주기신호를 생성하는 기구를 만들고 이것을 내부모델로 해서 폐루프내에 만들어서 주기의 임의의 주기함수는 한 주기에 대응하는 임의의 초기함수를 가하여 이것을 기억시켜 놓은 뒤 주기마다 반복시켜서 나오는 것에 의해 생성될 수 있다.

이러한 주기함수 발생기를 반복보상기(Repetitive compensator)라고 하고 반복보상기를 이용하여 구성한 제어계를 반복제어계(Repetitive control system)라고 한다.

아래의 Fig. 3.5은 반복제어계의 블록다이어그램을 나타낸 것으로 $P(s)$는 플랜트를 나타내며, $C(s)$는 제어기의 안정화를 위한 보상기 그리고 $F(s)$는 주기적인 입력의 필터링을 위한 필터이다. 본 연구의 반복제어계에서 반복보상기는 비례게인만 사용하였고 필터링은 고려하지 않았다.

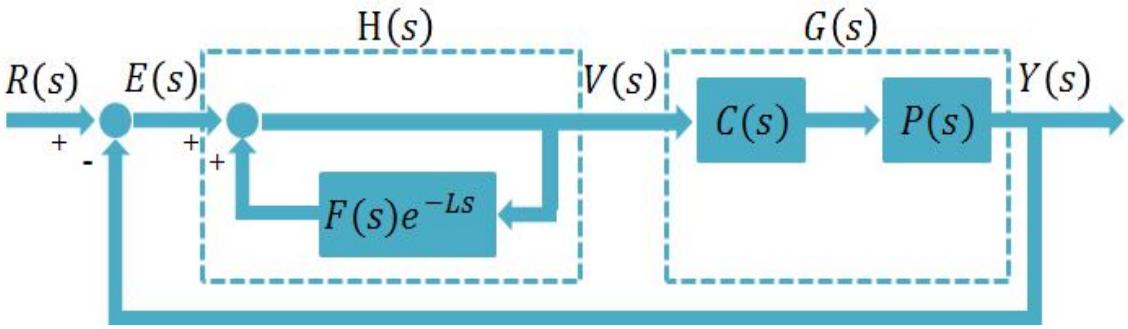

그림 3.5 반복제어 시스템의 블록다이어그램

Fig. 3.5 Repetitive Control system's Block diagram

3.2.3 반복제어계 시뮬레이션

반복제어계 시뮬레이션에서는, 우선 첫 번째 루프에서는 초기에 설정된 값을 이용하여 계산을 하도록 하며 이러한 경우 대부분 초기의 값은 '0'으로 되고 있다. 각 루프를 행할 때에는, 각 제어 샘플링 주기에 맞추어 전체 데이터 베이스를 구성하고, 이를 반복하여 한 루프 동안 시뮬레이션을 행하도록 한다.

첫 번째 사이클의 주행이 끝난 결과와 이때 전 샘플링 주기에 이용된 저장 Data(Input data, Output data and error data)를 통해 해당 주기의 입력 및 피드백 값을 조절하여 새로운 데이터를 생성하며, 이와 같은 흐름을 반복하여 전체 루프에서의 제어를 행하도록 한다.

반복제어계의 주된 원리는 반복적으로 이전 주기의 정보 즉, 입력, 출력 및 오차 값들을 통해 현재의 제어입력을 결정하여 시스템으로 들어가는 신호를 조절하는 것이다. 이때 반복적으로 측정되어진 출력값의 최대치가 목표치와 가까운 출력이 나타나게 되면 오차가 거의 없는 것이 되어 반복제어는 완료가 되고 해당 제어계는 상태를 유지하여 일정한 출력을 나타내게 된다.

반복제어계를 이용하여 5회 경과된 이후의 시뮬레이션 결과는 그림 3.6에서 확인할 수 있으며, 1회에 발생하였던 오버슈트가 없어지는 시점까지 제어를 수행하고 그 이후의 동작은 오버슈트가 없이 안정한 출력이 나오는 것을 알 수 있다.

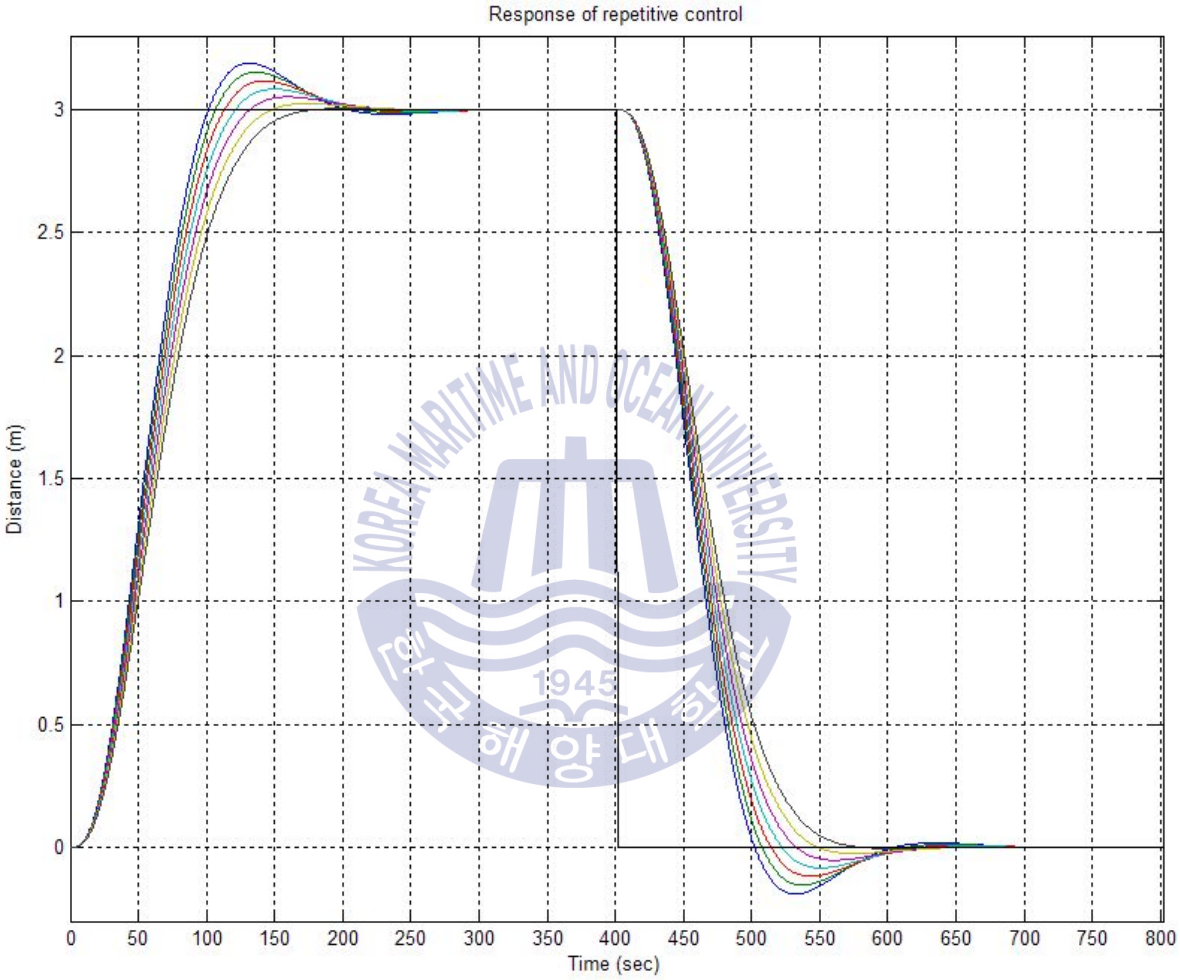

그림 3.6 반복제어의 응답결과

Fig. 3.6 Response of repetitive control

3.3 동기화 제어 알고리즘

3.3.1 동기화 제어 알고리즘의 필요성

물류센터에서는 작업 스케쥴에 의하여 여러 가지 상태의 화물 적재가 이루어질 수 있다. 사용자가 요구하는 이동랙의 크기도 해당 업종에 따라 다를 것이며 이동랙의 형태와 적재되는 적재물의 무게 또한 달라질 수 있다. 또한 상황에 따라 여러 개의 이동랙을 동시에 움직여야 할 수도 있다.

이때 각 이동랙이 내부적인 제어를 통해 고정된 제어 과정을 거친다면 각 이동랙의 파라미터 변화가 서보제어계로 제어가 불가능할 수 있다. 예를 들면 빈 이동랙과 화물이 모두 차 있는 이동랙은 당연하게 화물하중의 차이로 인해 구동부의 동일한 토크에도 속도 차이가 날 수 밖에 없다. 따라서 각 이동랙이 일정한 오차 범위 내에서 같이 움직일 수 있도록 동기화 제어 알고리즘이 필요하다.

3.3.2 동기화 제어 알고리즘의 구현 방법

이번 동기화 제어 알고리즘에서는 각 상황에 따라 동기화의 주체가 될 Master를 정하게 되고, 정해진 Master의 주행경로를 따라 나머지 Slave들이 자체적으로 속도를 조정하여 이동하게 되도록 하는 것이다. 초기 이동은 메인 시스템의 이동명령에 따라 이동지시를 받은 이동랙들이 모두 움직이게 하며, 이때 각 이동랙의 동특성을 파악하는 기본 자료로 이용된다.

위의 초기 구동에서 이동된 데이터를 통해 Master와 Slave를 정하게 되는데 동일한 입력 즉, 일정한 토크가 구동부에 주어져도 이동한 거리(구동휠 간의 동기화는 각 휠의 회전각도)가 작은 이동랙이 있다면 이는 적재 화물의 무게가 많이 나가거나, 기계의 내부적인 요소 그리고 주변 환경 등의 외부적인 요소로

인해 속도(구동휠 간의 동기화에서는 각 휠의 각속도)가 평균보다 낮아진 것으로 이를 Master 이동랙으로 지정한다.

그리고 Master와 각 이동랙의 거리 차이(구동휠 간의 동기화에서는 각 휠의 회전각도) 즉, 오차를 구하고 이 오차를 위에서 고려한 제어계를 이용하여 적용한다. 하중 및 부하가 작아진 부분이 위의 시뮬레이션에 고려된 하중 및 부하(이번 연구에서 고려된 최대치)가 적용된 부분보다 빠른 속도를 가지므로 이를 조절하면 적절한 동기화제어가 가능하다.

3.3.3 동기화 제어 알고리즘의 구성 및 설계

본 연구에서 동기화 제어의 기본적인 알고리즘은 다음의 플로우 차트와 같이 나타낼 수 있다.

우선 기본 이동랙의 동작이 시작되면 그와 동시에 엔코더의 데이터 값을 수집한다. 그리고 수집된 데이터를 통해 첫 번째로 계산될 이동랙의 좌우측 구동휠 동기화를 진행한다. 이동거리의 차이를 통해 Master와 Slave를 설정하고 Slave가 Master를 따라가는 방식을 가진다. 그리고 이때 반복제어가 동시에 적용이 되는데 첫 번째 주기에서의 오차를 저장하여 다음 주기의 동작 제어에 적용한다.

이렇게 첫 번째 이동랙의 구동휠 동기화가 진행이 되고 그와 동시에 다른 이동랙의 제어부에서도 해당 동기화, 즉 구동휠 간의 동기화 제어가 진행될 것이다. 그 후 각 이동랙의 구동휠 동기화 제어가 완료되면 각 이동랙들의 데이터를 이용해 다중 이동랙 간의 동기화 제어를 실행한다.

다중 이동랙 간의 동기화 제어에서도 반복적으로 제어가 적용되며 이동랙간의 첫 번째 주기 오차를 저장하고 그 다음 주기에 해당 오차를 이용해 동기화 제어에 적용하여 전체적인 주행 제어를 수행하게 된다.

다음 각 파트에서는 좌우 양측의 구동휠에 다른 부하가 작용한 것처럼 파라미터 값을 조정하여 시뮬레이션을 적용하는데 각각의 상태를 관찰하고 이의 동

기화제어를 진행한다. 그리고 그 다음 순서인 모듈별 동기화제어 역시 진행한다.

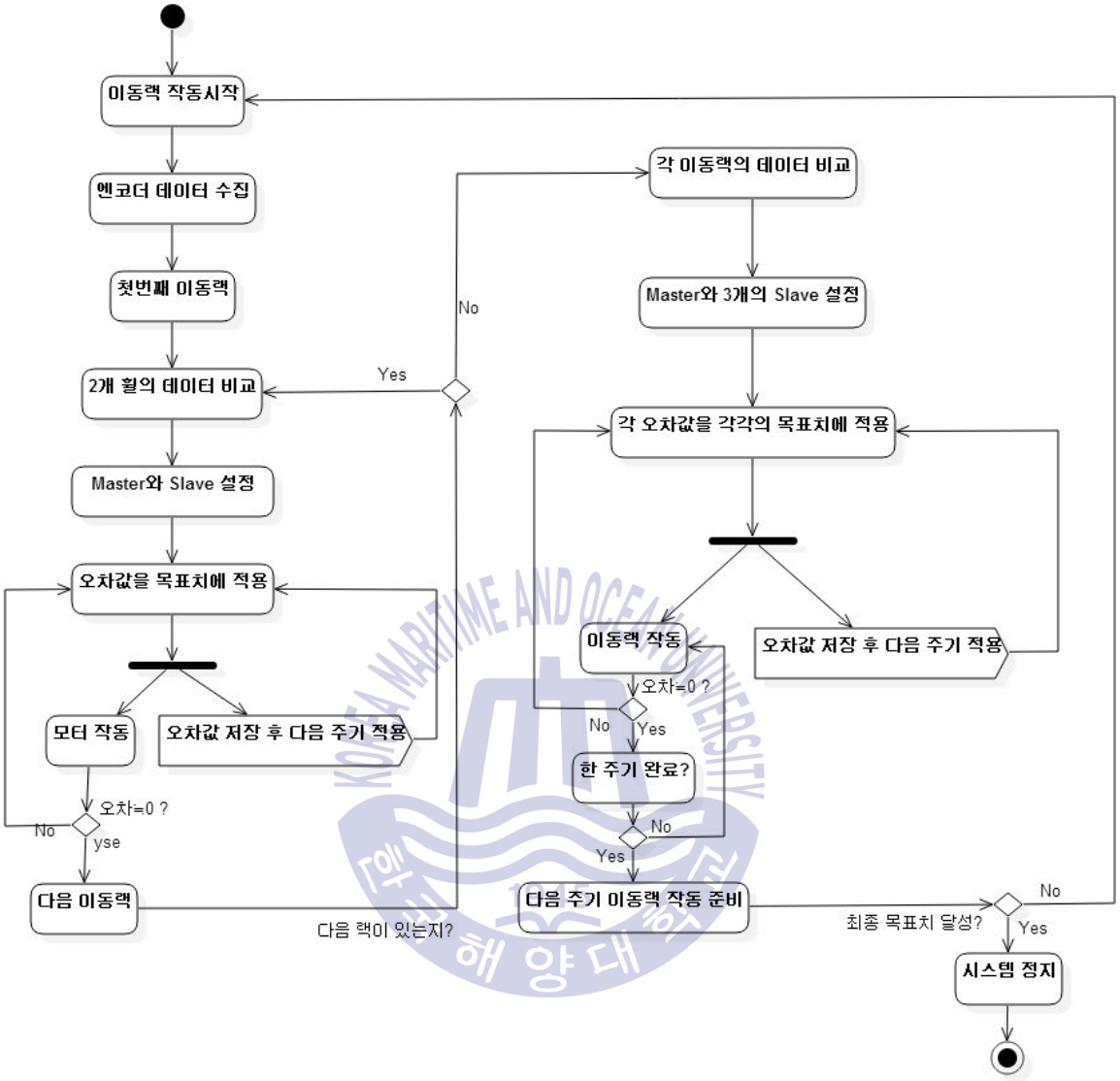

그림 3.7 동기화 제어 알고리즘

Fig. 3.7 Control algorithm for synchronize

모듈별 동기화제어는 모두 다른 상태를 가지고 있는 이동랙에 대하여 시뮬레이션을 하는데 이때 각각의 데이터를 수집하고 이를 분석하여 각 이동랙 간의 위치가 일정하고 오차가 많이 발생하지 않도록 제어를 하며 이를 통해 동기화 제어의 유효성을 확인하고자 한다.

3.4 이동랙 구동휠 동기화 제어

전 절에서 설명한 구동휠 간의 동기화제어 알고리즘을 통하여, 단일 이동랙의 주행 결과가 정해진 주행 궤적을 추종할 수 있도록 제어하고자 한다. 해당 결과는 MATLAB의 시뮬레이션을 통해서 동기화 제어를 적용하기 전과 후로 나누어 결과 데이터를 확인한다.

3.4.1 이동랙 구동휠 동기화 제어기 구성

1차적으로 동기화 제어에서는 작업에 따라 이동이 진행 중인 두 개 구동휠 사이의 오차를 측정하여 제어에 적용한다. 동일한 스펙의 구동부에 같은 입력을 넣어주어도 오차가 발생하게 되는데 하나의 이동랙에서 좌측과 우측 구동휠의 이동결과는 이동랙의 기계적 구조 혹은 적재 화물의 편 하중에 의해 다르게 나타날 수 있다.

이동랙의 주행이 시작되었을 때 각 구동휠의 데이터를 수집하고 이를 비교하여 그 오차값을 제어하는데 사용한다. 데이터 수집결과 좌우 특성 차이로 인해 속도가 빠른 쪽과 느린 쪽으로 구분이 될 텐데 S/W 상으로도 최대치 입력을 넣고 H/W 상으로도 전압을 더 크게 넣어주지 않는 이상 속도를 높일 수는 없을 것이다.

따라서 속도가 빠른 쪽을 느린 쪽에 맞춰서 갈수 있게끔 동기화 제어를 수행한다. 기존의 서보 동작에 사용되는 입력과 출력사이의 오차에 두 구동휠 사이에서 발생하는 오차를 적용하여서 빠른 쪽의 이동 속도를 조절한다. 이와 같은 동기화 제어를 적용하면 주행 중 두 개 구동휠 사이의 오차를 줄일 수 있다.

이러한 방식의 동기화 제어를 사용한다면 제어를 적용하기 전보다는 주행 궤적과 구동부의 작동 결과 사이의 오차가 많이 줄어들 것이다. 하지만 이것만으로는 주행 궤적과 최종 작동 결과를 정확히 일치시켜 최종 목표에 달성하기는 어려움이 있다 아무리 오차를 줄인다고 해도 줄여나가는 상황에서의 오차로 인

해 조금은 차이가 남아 있는 것이다.

하지만 동기화 제어를 적용하기 전보다는 차이가 작아질 것이며 반복적인 움직임을 수행하는 이동랙의 특징에 따른 제어 오차 해결 방법으로 반복 제어를 함께 적용하여 동기화 제어 알고리즘을 수행하면 조금의 차이도 해결할 수 있을 것이다.

3.4.2 시뮬레이션

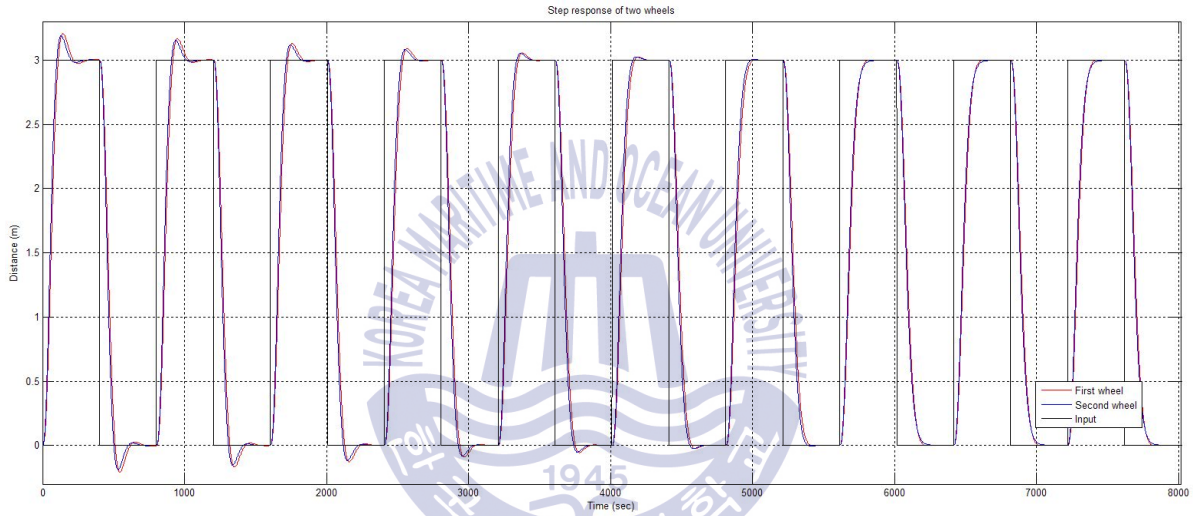

그림 3.8 동기화 제어 전 두 개 휠에서의 스텝 응답

Fig. 3.8 Step response of two wheels before using repetitive control

위의 그래프는 하나의 이동랙 즉, 두 개의 휠을 고려하여 시뮬레이션한 MATLAB 결과 그래프로 1[m]의 이동 입력을 준 후 결과값이다. 이때 한쪽은 적재되는 화물의 파라메터 값을 임의로 조절하여서 좌우의 부하가 다른 상태를 시뮬레이션에 적용하였으며 이것은 그래프를 보면 알 수 있듯이 결과값에 변화를 주었다.

이동랙 좌우 각각의 구동휠에 작용하고 있는 힘이 다르다면 제어부에서 전달되어지는 같은 입력 신호에도 구동부의 출력이 달라질 수 있다. 이러한 현상은 이동랙의 물리적 요인, 화물의 편하중 그리고 설치되어진 작업 환경에 따라 조

금씩 차이가 발생하며 이동랙의 주행방향을 틀어지게 만들어 정해진 주행 궤적을 따라가지 못하는 현상을 발생시킨다.

그래프를 보면 확인할 수 있듯이 시뮬레이션 결과로는 반복제어를 적용하고 있기 때문에 주기를 반복하면 할수록 입력값에 가까워지는 결과를 나오기 때문에 최종적으로는 두 구동휠 사이의 오차가 많이 줄어든 것처럼 보이지만 자세히 확인해보면 최종 결과값이 비슷할 뿐 중간 과정에는 여전히 오차가 존재한다.

따라서 앞서 설명했던 내용처럼 이동랙의 작동이 시작되고 그에 따른 최초 결과값으로 Master와 Slave를 설정한다. 그 후 각 구동휠 간의 이동거리 차이 데이터를 실시간으로 저장하면서 구동부에 적용한다. 이러한 방식으로 각 주기 안에서는 구동휠간의 오차를 줄이고 반복제어를 통해 결과값과 입력값 사이의 오차도 줄여나간다.

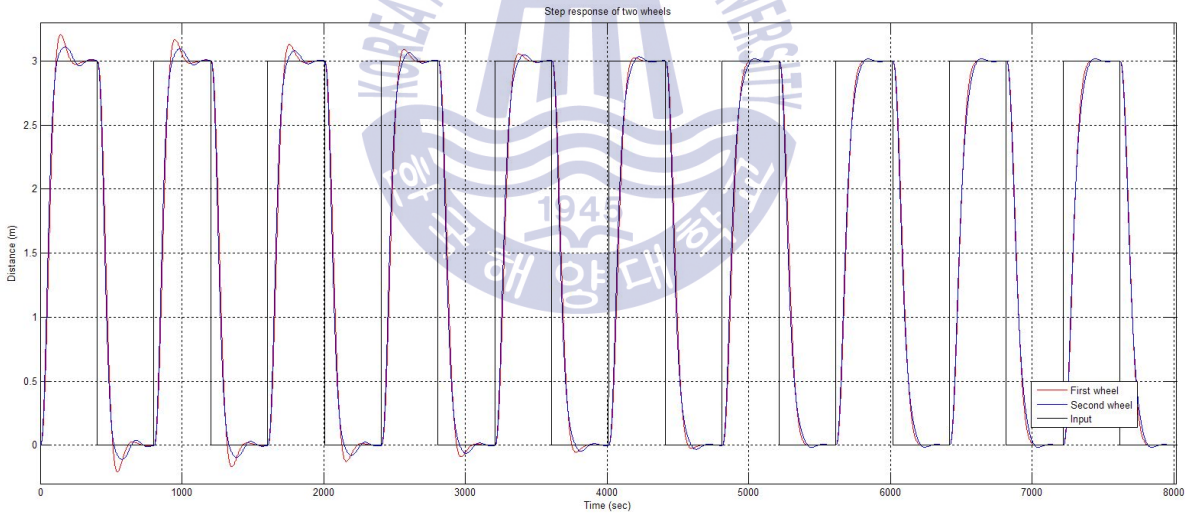

그림 3.9 동기화 제어 후 두 개 휠에서의 스텝 응답

Fig. 3.9 Step response of two wheels after using repetitive control

Fig 3.9을 확인해 보면 초기 동작에서의 데이터는 부하가 작은 쪽이 조금 앞서는 듯 보이지만 동기화 제어 알고리즘을 통해 속도를 맞춰나가는 것을 확인할 수 있었다. 그 과정에서 속도의 조절이 적정수준 보다 초과하게 되면 반대로 다시 속도를 조금 높여가며 Master 이동랙의 움직임에 맞춰가도록 제어 동

작을 수행하게 된다.

앞의 그래프만 보면 반복 제어에 의해 목표치에 적절하게 수렴하며 약간의 차이만이 결과로 나타나는 것을 볼 수 있다. 이를 명확히 확인하기 위해 이동랙 구동휠 간의 비교를 통해 시뮬레이션 결과를 확실히 확인하고자 한다.

우선 두 개 휠 사이의 데이터 차이 값을 제어하기 전과 후로 나누어서 데이터를 확인하고 분석해보았다. 그리고 각각의 결과는 다음 그림 3.10과 3.11과 같이 나타난다.

그림 3.10의 그래프는 동기화 제어를 적용하기 전에 계측한 두 개의 이동랙 구동휠 사이의 오차 그래프이다. 초기 오차가 발생한 상태로 구동이 진행되고 지속적인 반복 제어를 통해 구동부의 동작이 안정적으로 변하면서 두 개 이동랙의 오차가 조금씩은 줄어드는 것으로 확인된다.

그림 3.11의 그래프는 동기화 제어를 적용하고 난 후에 계측한 두 개의 이동랙 구동휠 사이의 오차 그래프이다. 동기화 제어를 적용하여도 동작 초기 오차가 조금 발생하게 된다. 이때에는 동기화 제어를 적용하기 전보다는 작은 오차를 가지고 시스템의 동작이 진행된다.

다음 두 개의 그래프를 비교해보면 반복 제어에 의해 주행 주기를 반복할수록 두 개의 이동랙 구동휠 사이의 오차가 줄어드는 것과 동작 초기에는 두 개의 출력사이에 오차가 존재한 다는 점에서 크게 차이가 나지 않는 것으로 확인할 수 있다.

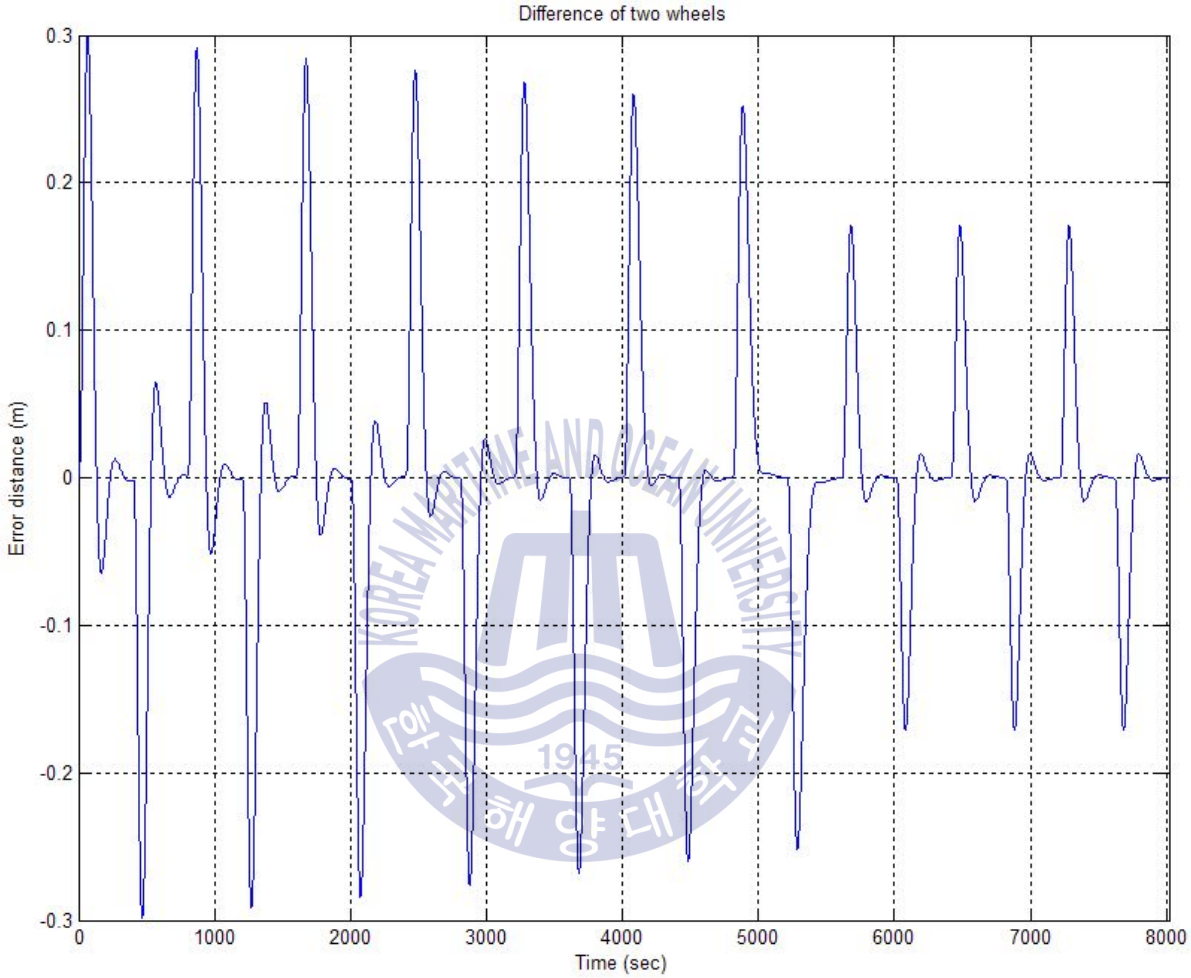

그림 3.10 동기화 제어 전 두 개 휠 사이의 차이

Fig. 3.10 Difference of two wheels before synchronized control

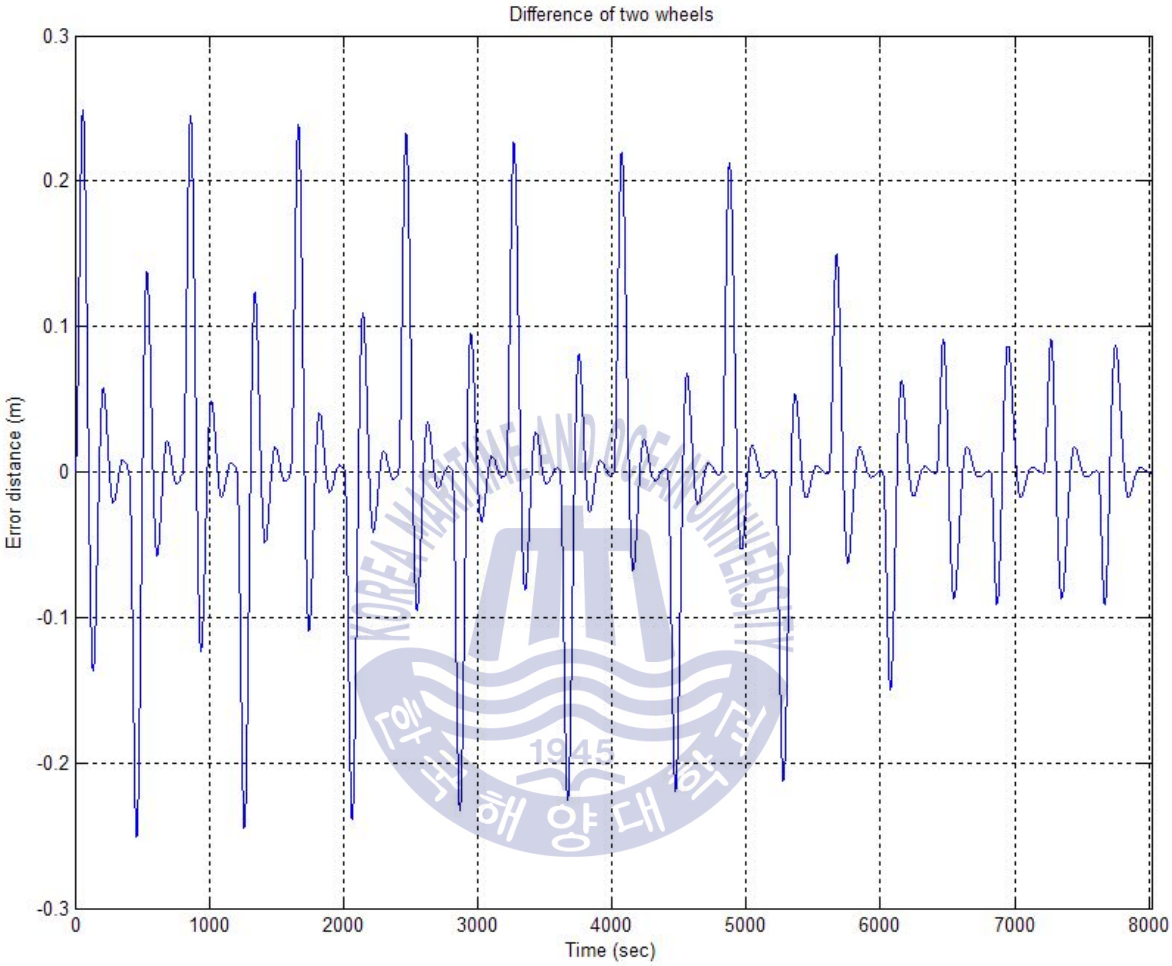

그림 3.11 동기화 제어 후 두 개 휠 사이의 차이

Fig. 3.11 Difference of two wheels after synchronized control

앞선 시뮬레이션에 따른 결과 그래프만으로는 동기화 제어의 효과를 확인하기 어려워 보인다. 따라서 다음으로는 이 오차값들의 누적치를 주기별로 나타내어 확인해보고자 한다.

다음의 그래프 그림 3.12와 그림 3.13은 이동랙 구동휠에 동기화 제어를 적용하기 전과 후로 나뉘어져 있으며 그래프의 출력 값은 앞서 설명한 것처럼 이동랙 구동휠의 오차 누적치이다.

그림 3.12의 그래프는 동기화 제어를 적용하기 전 오차값의 주기별 합계이다. 이 그래프에 따르면 어느 정도 반복 제어가 진행된 후의 주기부터는 반복 제어에 의해 구동이 안정하게 되므로 오차의 누적치 또한 줄어들게 된다.

그림 3.13의 그래프는 동기화 제어를 적용한 후 오차값의 주기별 합계이다. 단지 구동휠 간의 오차 값으로만 비교해서는 확인하기 어려웠지만 누적값을 보면 오차가 절반 가까이 줄어든 것을 확인할 수 있다. 그리고 동기화 제어를 반복 제어와 함께 주기마다 반복 적으로 제어를 수행하면서 이동랙의 동작이 거의 안정화된 마지막 주기 이후에는 아주 작은 오차만 남아 있는 것으로 확인할 수 있었다.

이렇게 이동랙 구동휠 간의 동기화 제어가 진행되고 모든 이동랙의 좌우 동기화가 맞춰지게 되면 동기화 제어 알고리즘에 따라 다른 다중 이동랙 간의 동기화 제어를 진행하게 된다. 따라서 앞선 내용과 같이 이동랙 구동휠 간의 동기화 제어 시뮬레이션을 진행하였고 다음으로는 다중 이동랙 동기화 제어를 시뮬레이션을 진행 한다.

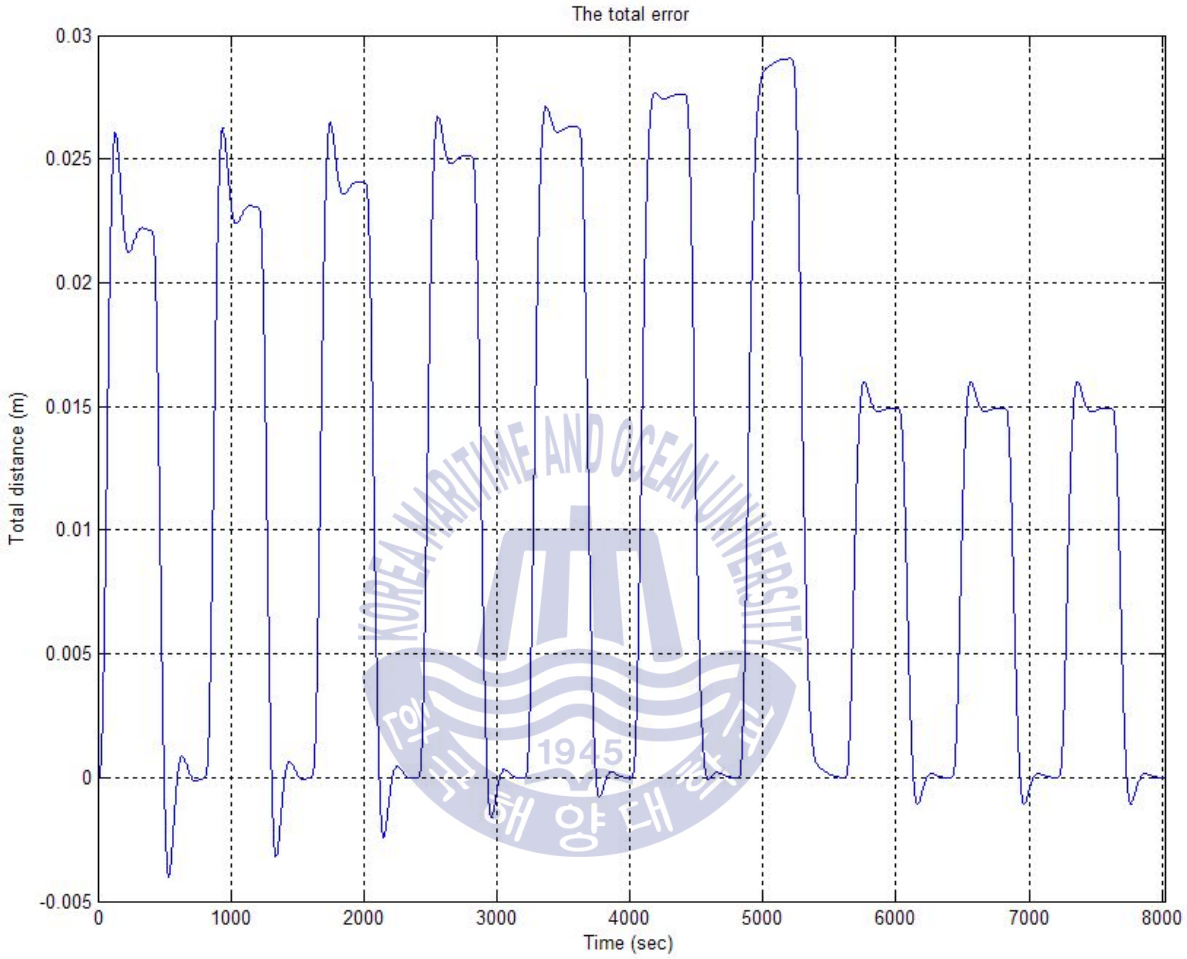

그림 3.12 동기화 제어 전 두 개 휠 사이의 차이 합

Fig. 3.12 The sum of difference between two wheels before using synchronization control

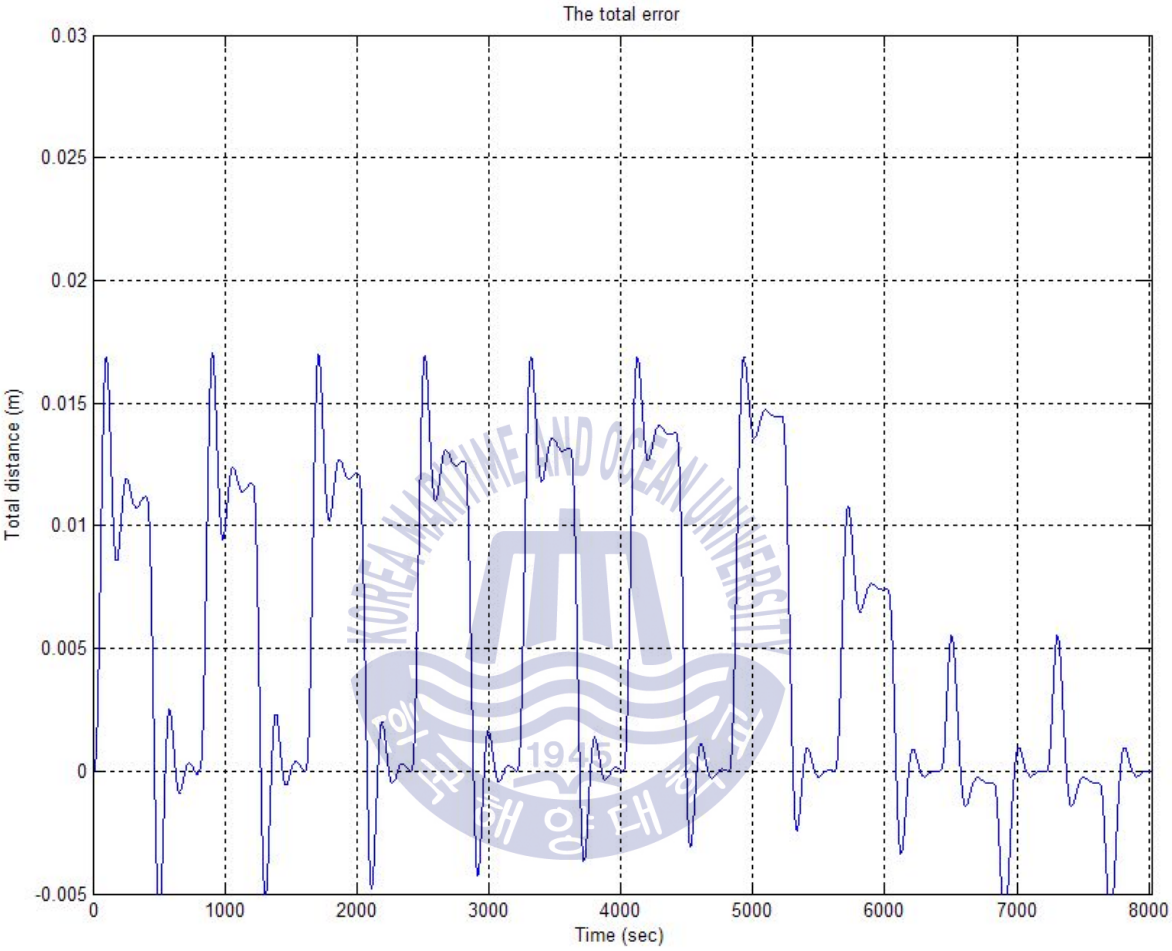

그림 3.13 동기화 제어 후 두 개 휠 사이의 차이 합

Fig. 3.13 The sum of difference between two wheels after using synchronization control

3.5 다중 이동랙 동기화 제어

모듈별 동기화 제어 역시 구동휠간의 동기화 제어와 크게 다른 것은 없다. 반복적으로 반복제어를 수행하며 동기화 제어 알고리즘을 같이 적용한다. 하지만 이때 다른점은 이동랙 구동휠 간의 동기화 제어는 두 개의 데이터를 동기화하면 되지만 다중 이동랙의 경우 이동 명령을 수행해야하는 이동랙 모두가 함께 동기화 제어를 적용한다. 자세한 내용은 다음과 같다.

3.5.1 다중 이동랙 동기화 제어기 구성

우선 구동휠간의 동기화가 진행되면서 두 휠의 이동 변위의 평균값을 해당 이동랙의 이동한 거리로 하고 이 데이터를 가지고 동기화 제어를 행한다.

예를 들어 4개의 이동랙이 움직인다고 가정하면 각 이동랙에 적재된 랙의 무게가 모두 다를 것이다. 물건이 적재되지 않아 부하가 거의 없는 상태일수도 있고 물건이 가득 차 부하가 최고치에 있을 수도 있다. 이러한 차이로 인해 이동랙의 속도 차이가 발생하는 것이다.

이러한 속도 차이에서 나오는 오차를 통해 다중 이동랙 사이에도 Master와 Slave를 설정한다. 물론 이 상황에서도 속도가 느린 이동랙을 Master로 설정하고 나머지 이동랙은 Slave가 된다. 구동휠과 마찬가지로 H/W의 한계를 넘어 더 큰 토크를 발생시킬 수 없기 때문에 부하가 적어서 속도가 빠른 이동랙이 속도가 느린 이동랙을 따라가는 방법이 적절하다.

이처럼 이동랙 간에도 속도를 맞춰나가며 오차를 줄인다면 이동랙에 작용하는 부하나 전체적인 상태가 크게 변하지 않는 이상은 오차가 거의 없이 작동될 것이다. 만일 변하게 되더라도 다시 반복적으로 동기화 제어를 수행하여 위치를 조정하게 된다.

3.4.2 시뮬레이션

우선 그림 3.14는 이동랙이 4개 있는 것을 가정하여 시뮬레이션 한 것으로 각각의 이동랙에 다른 무게의 화물에 적재되어 있는 것으로 파라메터를 설정하였다.

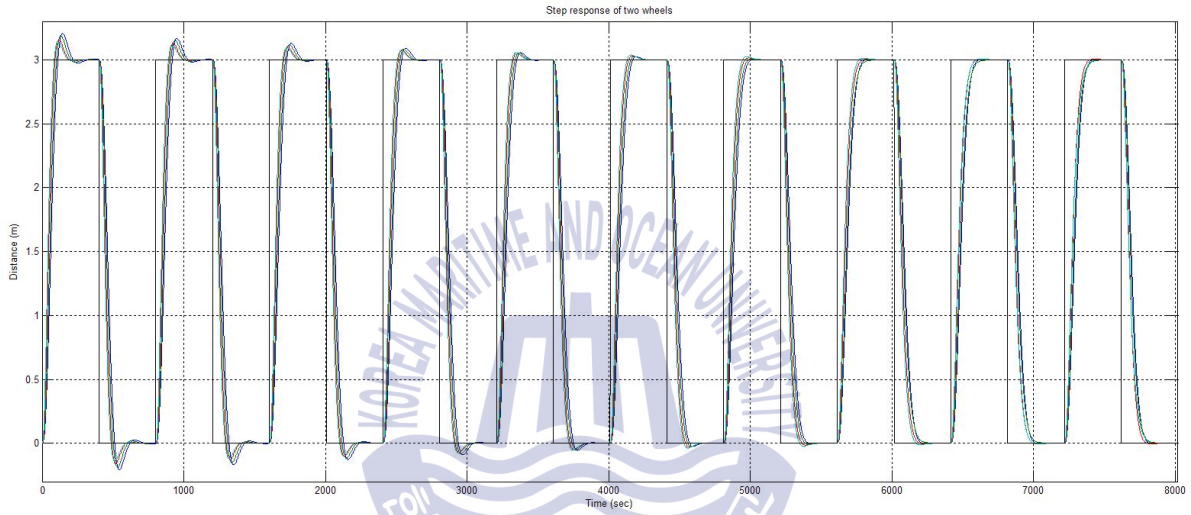

그림 3.14 동기화 제어 전 네 개 이동랙의 스텝 응답

Fig. 3.14 Step response of four mobile racks before using repetitive control

그림 3.14의 결과를 확인해보면 이동랙의 구동휠 간의 결과값과 유사하게 나타난다. 처음 동작에는 전체 하중이 가장 큰 이동랙의 속도가 늦게 나타나는데 이는 전체적으로 물건의 적재가 많이 되어있는 이동랙의 구동부에는 작용하는 부하가 크기 때문이다.

그리고 그 외의 이동랙은 처음 언급한 이동랙보다는 속도가 비교적 빠른데 이는 적재되어 있는 화물의 무게가 작게 설정되어 가장 무게가 많이 나가는 이동랙보다는 속도가 빠르게 나타난다.

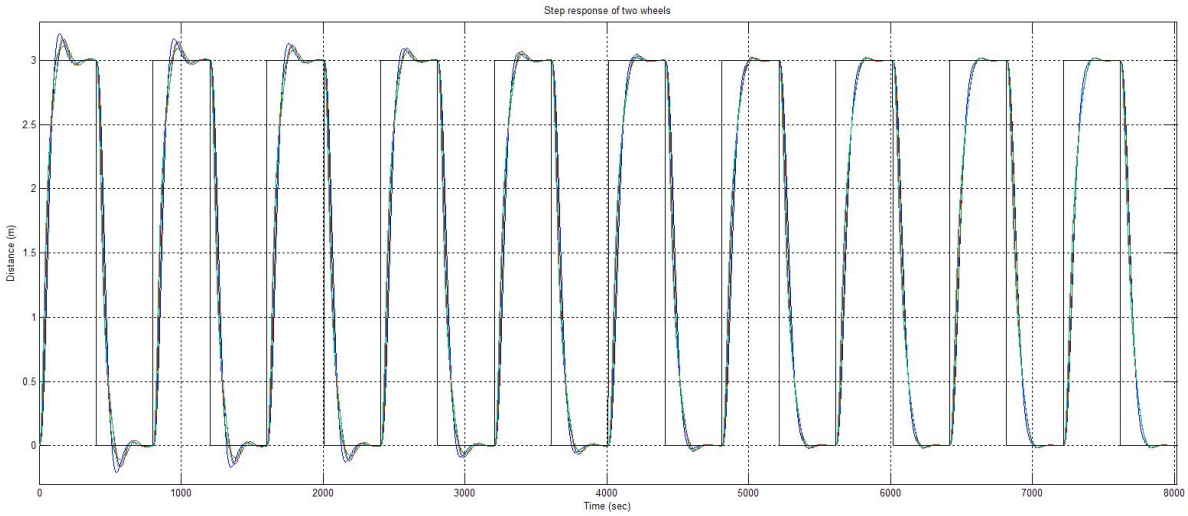

그림 3.15 동기화 제어 후 네 개 이동랙의 스텝 응답

Fig. 3.15 Step response of four mobile racks after using repetitive control

위의 그래프 3.15는 동기화 제어를 적용한 후의 다중 이동랙의 이동 결과 그래프이다. 이 결과 또한 구동휠 간의 동기화 제어와 비슷하게 나타나며 동기화 제어 알고리즘에 의해 구동중에도 지속적으로 속도를 조절해나간다.

우선 이동랙 구동휠 간의 동기화 제어가 진행되고 이어서 다중 이동랙 사이의 동기화 제어도 진행되는데 이동랙 사이에서도 Master와 Slave를 설정하여 동기화 제어를 진행한다.

적재된 화물에 따라 이동랙의 파라메터 특성이 달라지며 이로 인해 이동랙간의 속도차이가 발생한다. 이때 속도가 느린 이동랙을 Master로 설정하고 그 외에 비교적 빠른 속도를 가지는 이동랙을 Slave로 설정하며 동기화 제어를 진행한다.

다음 그래프는 이동랙간의 주행 오차 그래프로써 이것을 통해 Master의 속도에 맞춰 Slave가 속도를 조절하며 동작하여서 동기화 제어를 수행한 결과를 비교하고 분석해본다.

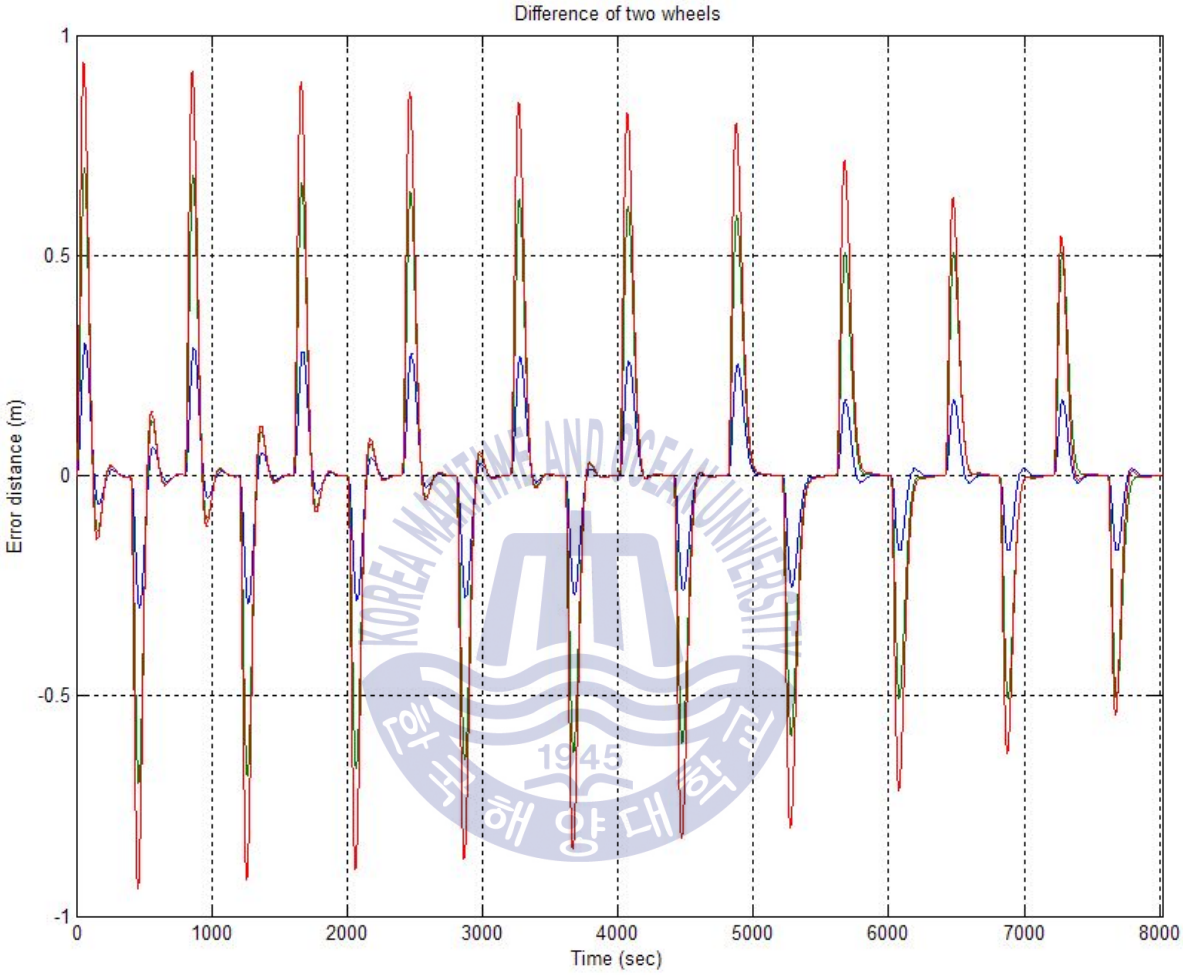

그림 3.16 4개 이동랙의 동기화 제어 전 오차 결과

Fig. 3.16 Difference of four mobile racks before synchronized control

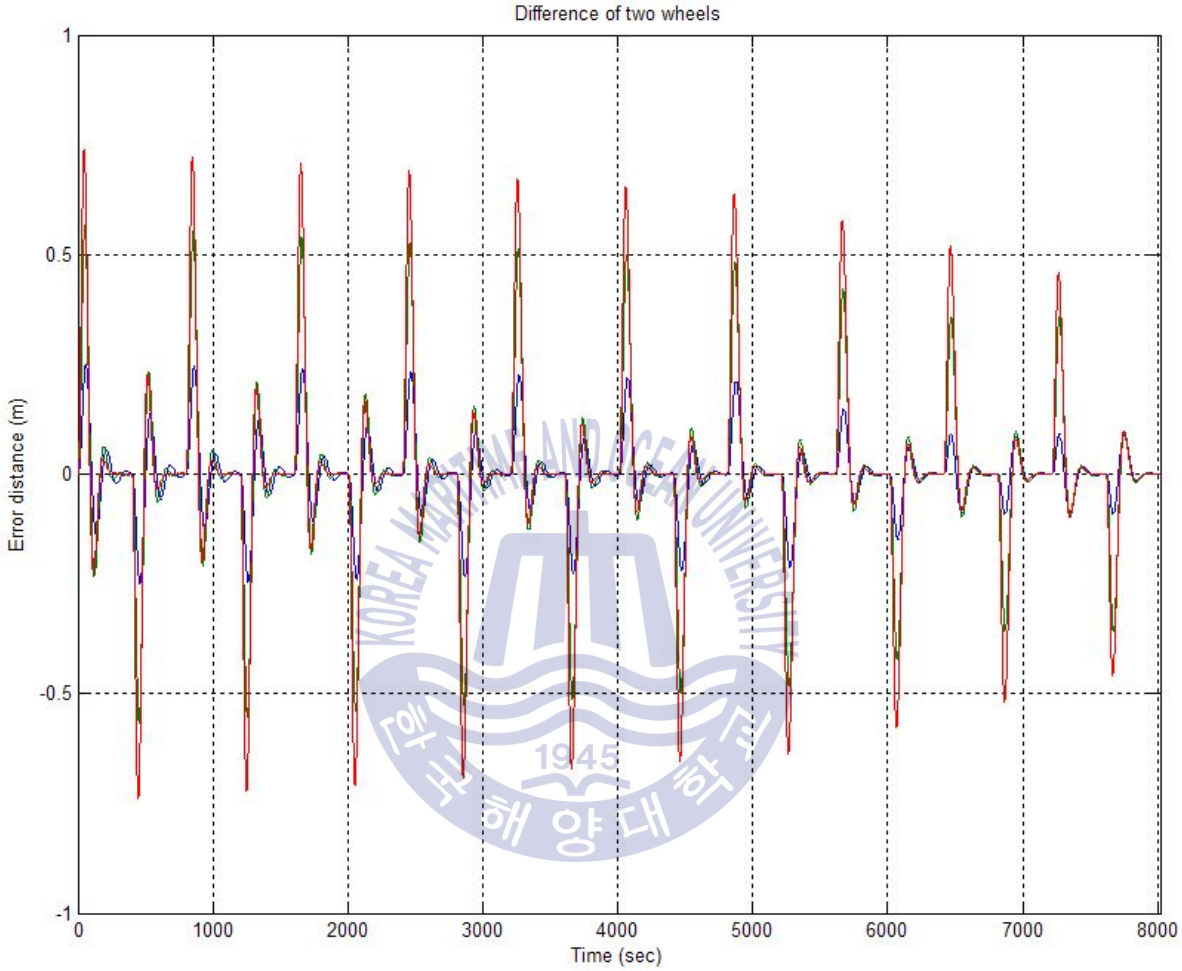

그림 3.17 4개 이동랙의 동기화제어 오차 결과

Fig. 3.17 Difference of four mobile racks after synchronized control

우선 네 개의 모듈 사이의 데이터 차이 값을 제어하기 전과 후로 나누어서 데이터를 확인하고 분석해보았다. 그에 따른 결과 그래프는 앞서 나와 있는 그림 3.16과 그림 3.17이다.

앞의 그림 3.15의 그래프는 동기화 제어를 적용하기 전에 계측한 네 개의 이동랙 사이의 오차 그래프이다. 초기 오차가 발생한 상태로 구동이 진행되고 지속적인 반복 제어를 통해 이동랙의 구동이 안정적으로 변하면서 나머지 이동랙과의 오차도 조금씩은 줄어드는 것으로 확인된다.

그림 3.17의 그래프는 동기화 제어를 적용하고 난 후에 계측한 네 개의 이동랙 사이의 오차 그래프이다. 동기화 제어를 적용하여도 동작 초기 그리고 각각의 주기가 진행됨에도 불구하고 오차가 발생하는 것으로 나타난다.

하지만 동기화 제어를 적용하기 전과 후, 두 개의 그래프를 전체적으로 비교해보면 동기화 제어를 적용하기 전보다는 작은 오차를 발생하고 있는 것으로 나타난다.

이번 시뮬레이션에서도 이동랙 구동휠 간의 동기화 제어에서 확인했던 것처럼 유사한 결과가 나오는데 반복적인 제어로 인해 몇 주기 후에는 오차가 줄어들지만 동기화 제어를 적용하기 전과 후를 비교해보면 오차가 조금 감소하여 크게 줄어들지는 않는 것으로 보여진다.

따라서 이를 명확히 확인하기 위해 다중으로 이루어진 이동랙 사이의 오차값을 이용해 주기별로 누적치를 계산하고 이를 비교하여 시뮬레이션 결과를 확실히 확인하고자 한다.

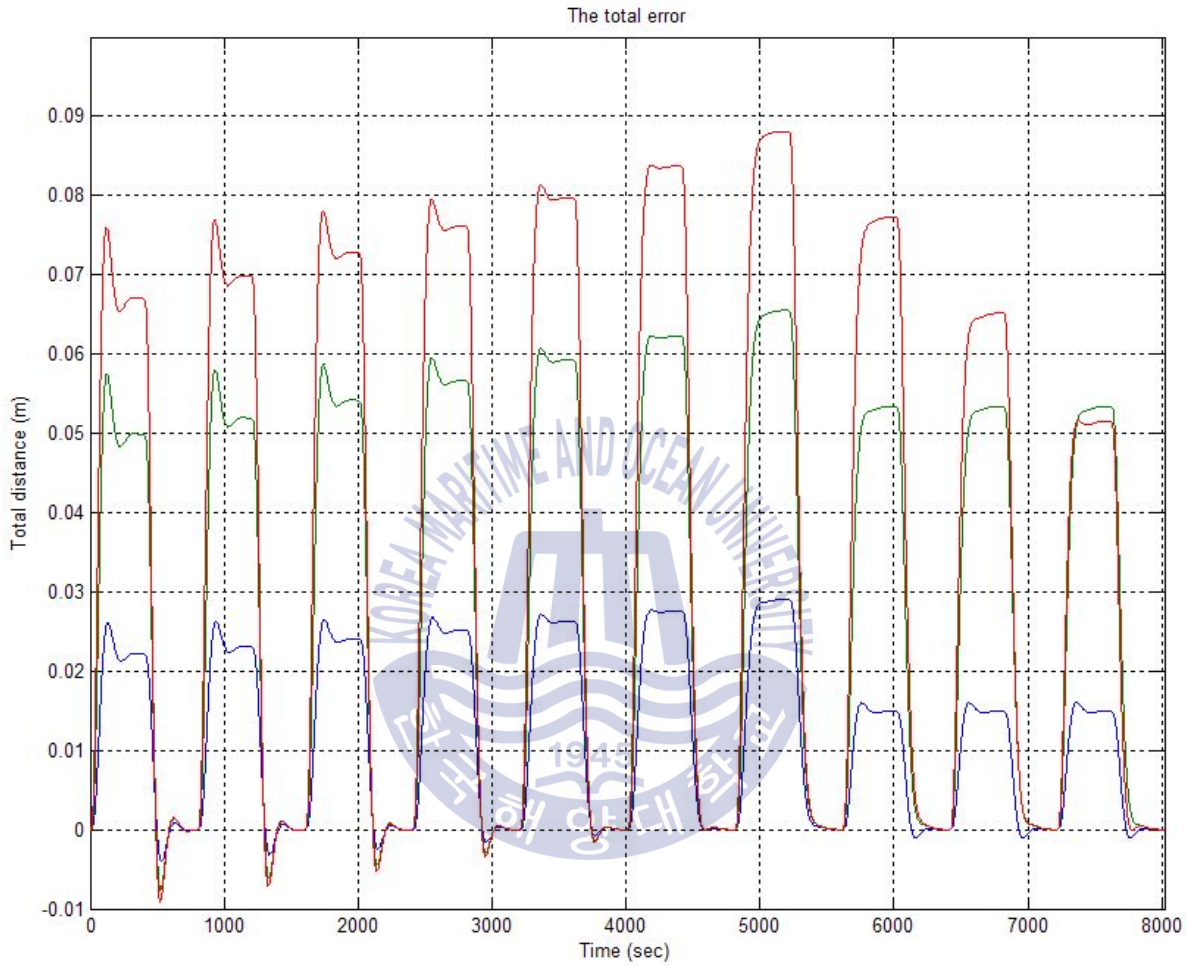

그림 3.18 동기화 제어 전 네 개 이동랙 사이의 차이 합

Fig. 3.18 The sum of difference between four mobile racks before using synchronization control

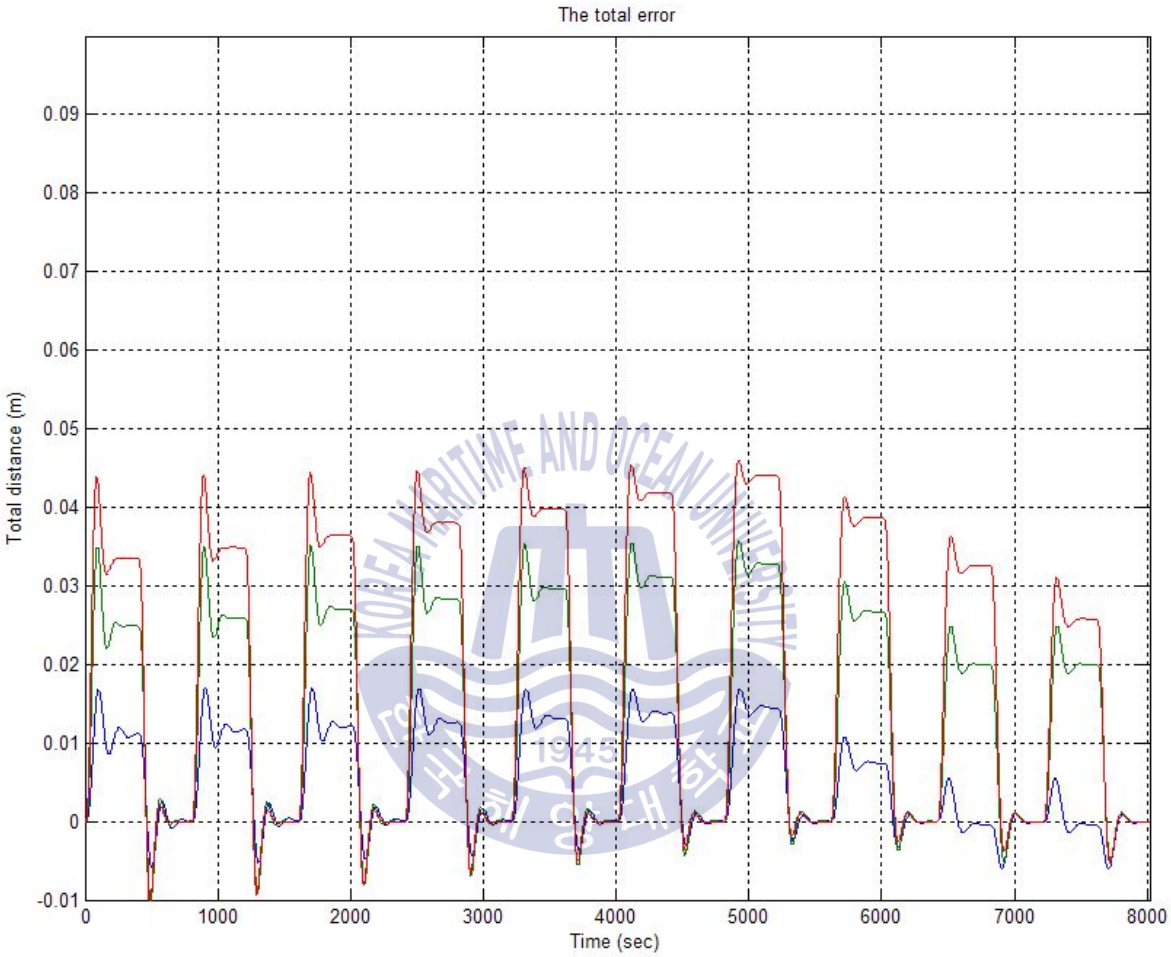

그림 3.19 동기화 제어 후 네 개 이동랙 사이의 차이 합

Fig. 3.19 The sum of difference between four mobile racks after using synchronization control

앞선 그래프의 그림 3.18과 그림 3.19은 이동랙의 모듈 사이에 동기화 제어를 적용하기 전과 후로 나뉘어져 있으며 그래프의 출력 값은 다중으로 된 이동랙 사이에서의 오차 누적치이다.

그림 3.18의 그래프는 동기화 제어를 적용하기 전 오차값의 주기별 누적치로 이 그래프에 따르면 어느 정도 반복 제어가 진행된 후의 주기부터는 반복 제어에 의해 구동이 안정하게 되므로 오차의 누적치 또한 줄어들게 된다.

그림 3.19의 그래프는 동기화 제어를 적용한 후 오차값의 주기별 누적치로 단지 이동랙 사이의 오차 값으로만 비교해서는 확인하기 어려웠지만 모듈별 동기화제어 역시 누적값을 확인해 보면 동기화 제어로 인해 오차가 절반 가까이 줄어든 것을 확인할 수 있다.

그리고 동기화 제어를 반복 제어와 함께 주기마다 반복 적으로 제어를 수행하면서 이동랙의 동작이 거의 안정화된 마지막 주기 이후에는 아주 작은 오차만 남아 있는 것으로 확인할 수 있었다.

이렇게 이동랙 구동휠 간의 동기화 제어 후 이동랙 사이의 동기화 제어까지 진행되면 모든 이동랙의 동기화가 맞춰지게 되며 모듈별로 구성되어 있는 무궤도 형식의 이동랙이 일정하게 움직이게 되어서 무궤도 형식에 의한 단점을 보완할 수 있게 된다.

다음으로는 시뮬레이션 한 알고리즘과 결과를 토대로 테스트용 모듈에 적용하여 반복제어계를 이용한 동기화 제어의 유효성을 확인해본다.

제 4 장 동기화 제어기 실험 및 고찰

이번 4장에서는 3장에서 시뮬레이션한 제어 기술을 테스트모델에 적용하여 시뮬레이션 결과를 증명하려 한다. 다음으로 이동랙 모델에 대해서 간략하게 설명하고 해당 실험 결과에 대한 고찰을 진행한다.

4.1 이동랙 모델의 개요

3장에서 확인한 시뮬레이션의 경우 현장에서 사용되어질 실제 모델의 특징과 상태를 고려한 것으로 해당 크기의 모델을 제작해서 작동 및 제어 테스트를 진행하기는 어렵다.

따라서 실제 모델에 적용되는 구성 중 일부 장치(안전장치, 전장설비 및 기타 구성품)는 본 연구에 불필요하므로 제외하고 시뮬레이션을 적용해 볼 수 있는 간단한 이동랙 모델을 고려해본다.

장치의 구성은 필요한 부분만 적용하며 이때 각 부분의 크기를 고려하면서 이동랙 모델의 크기를 최소한으로 할 수 있는 비율로 정하게 되었고 실제 모델보다 1/8정도의 크기로 설계하고 제작한다.

4.2 다중 이동랙 실험장치 구성

실험장치의 대략적인 구성은 아래의 시스템 아키텍쳐와 같다. 크게 PC, Arduino Due 그리고 Actuator로 나뉘어 진다. Actuator는 이동랙에 포함된 구동부들을 지칭하며 이를 컨트롤하는 것은 Arduino Due이고 PC를 이용해

Arduino Due를 제어한다.

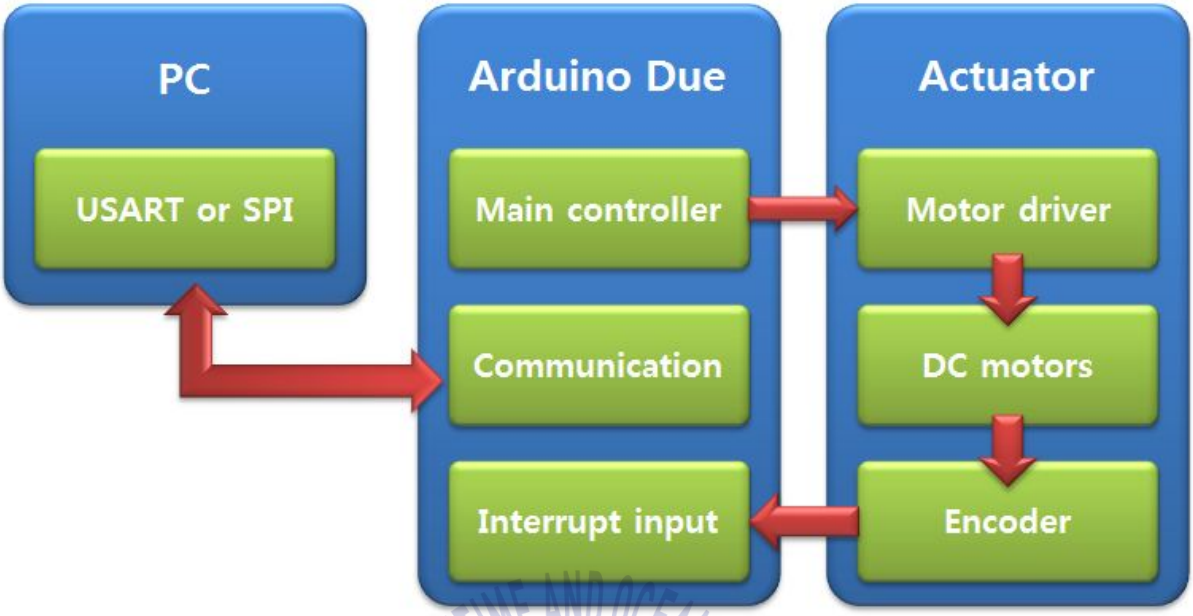

그림 4.1 시스템 아키텍쳐
Fig. 4.1 System architecture

PC의 기능은 Arduino Due를 제어하여 이동랙 구동에 작용하며 작동 결과로 나타나는 데이터 값을 통신을 통해 받게 된다. 이때 사용된 통신 방법은 Xbee를 이용한 무선통신으로 각 이동랙의 통신 파트에서 각각 모터의 엔코더 데이터를 수집하여 전송하는데 사용된다.

Arduino Due는 PC의 명령에 따라 이동랙에 제어 신호를 보내게 되는데 Analog Output을 통해 모터 작동 PWM을 출력하고 Digital Output을 통해 모터가 구동될 방향에 대한 정보를 출력하여 모터를 작동시킨다.

Digital Input에 인터럽트을 적용하여 모터에서 나오는 엔코더 데이터를 수집하는데 방향에 대한 정보는 모터 드라이브로 전송되는 방향 데이터가 존재하므로 로터리 엔코더의 A, B상 두 개중에 A상 하나만 인터럽트로 데이터를 받았다.

다음 그림은 테스트 모듈을 구성하는 회로도와 도면이다.

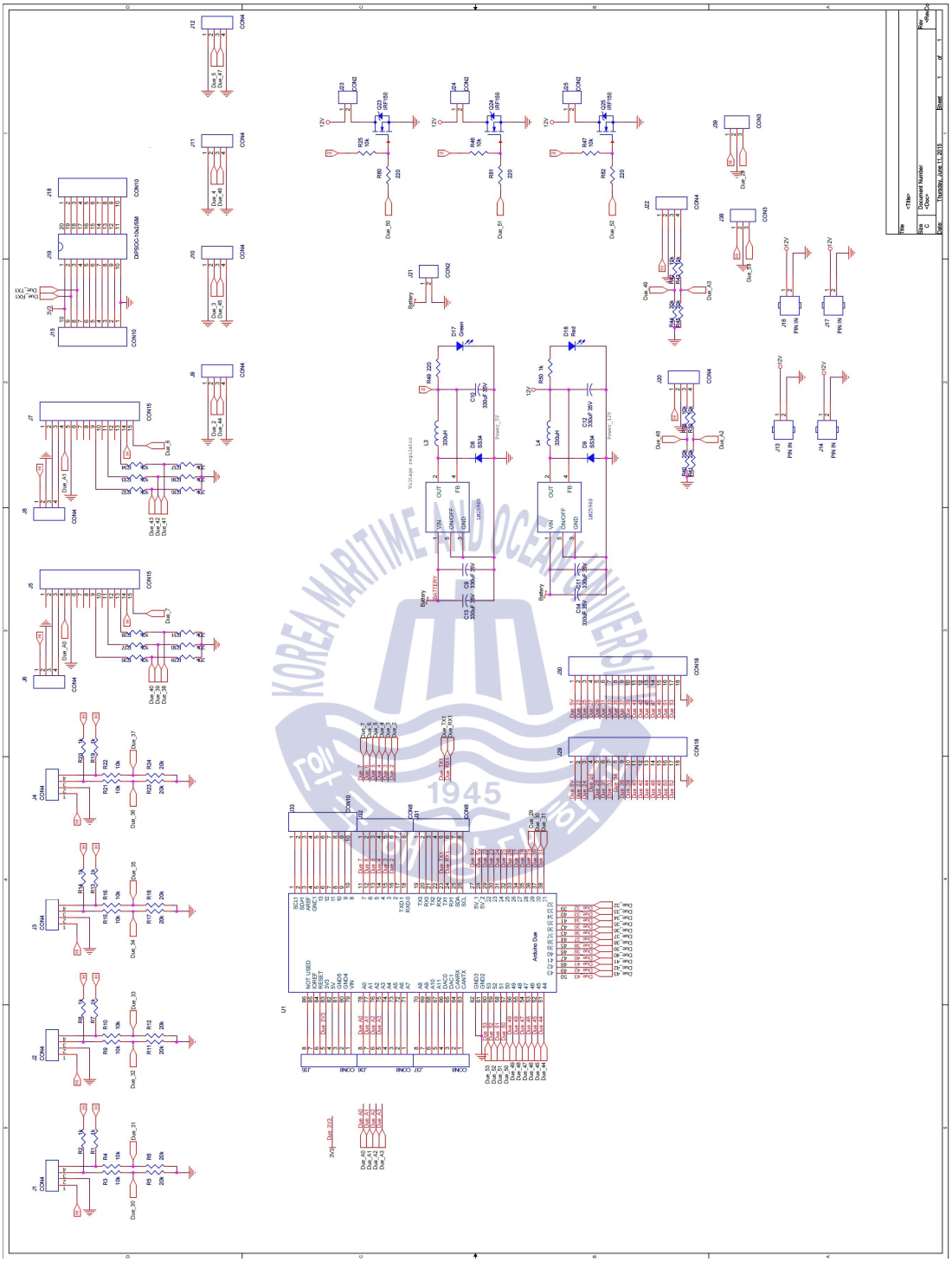

그림 4.2 테스트 모듈의 회로도

Fig. 4.2 Circuit of test module

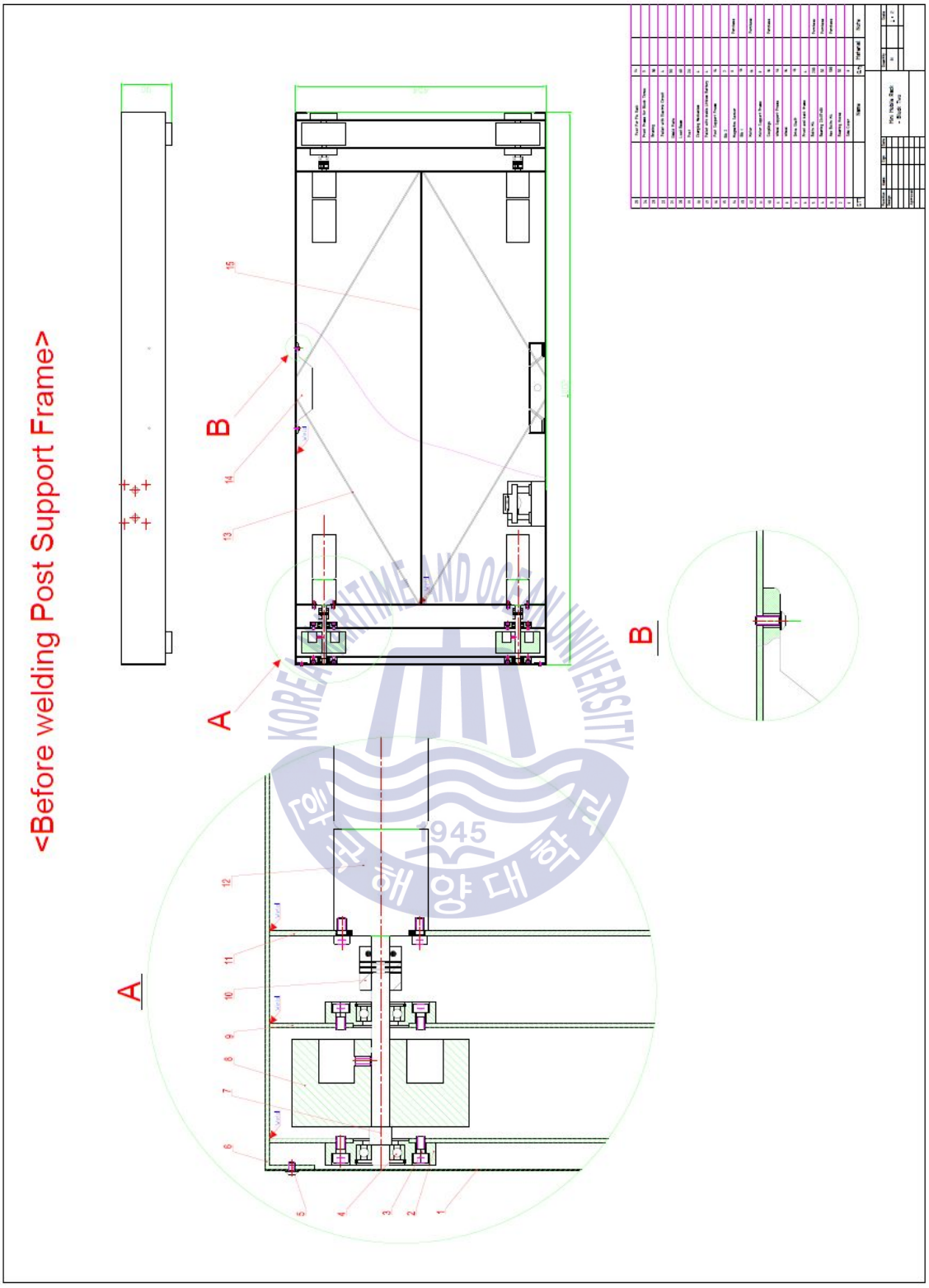

그림 4.3 테스트 모듈의 도면 1

Fig. 4.3 Drawing of test module 1

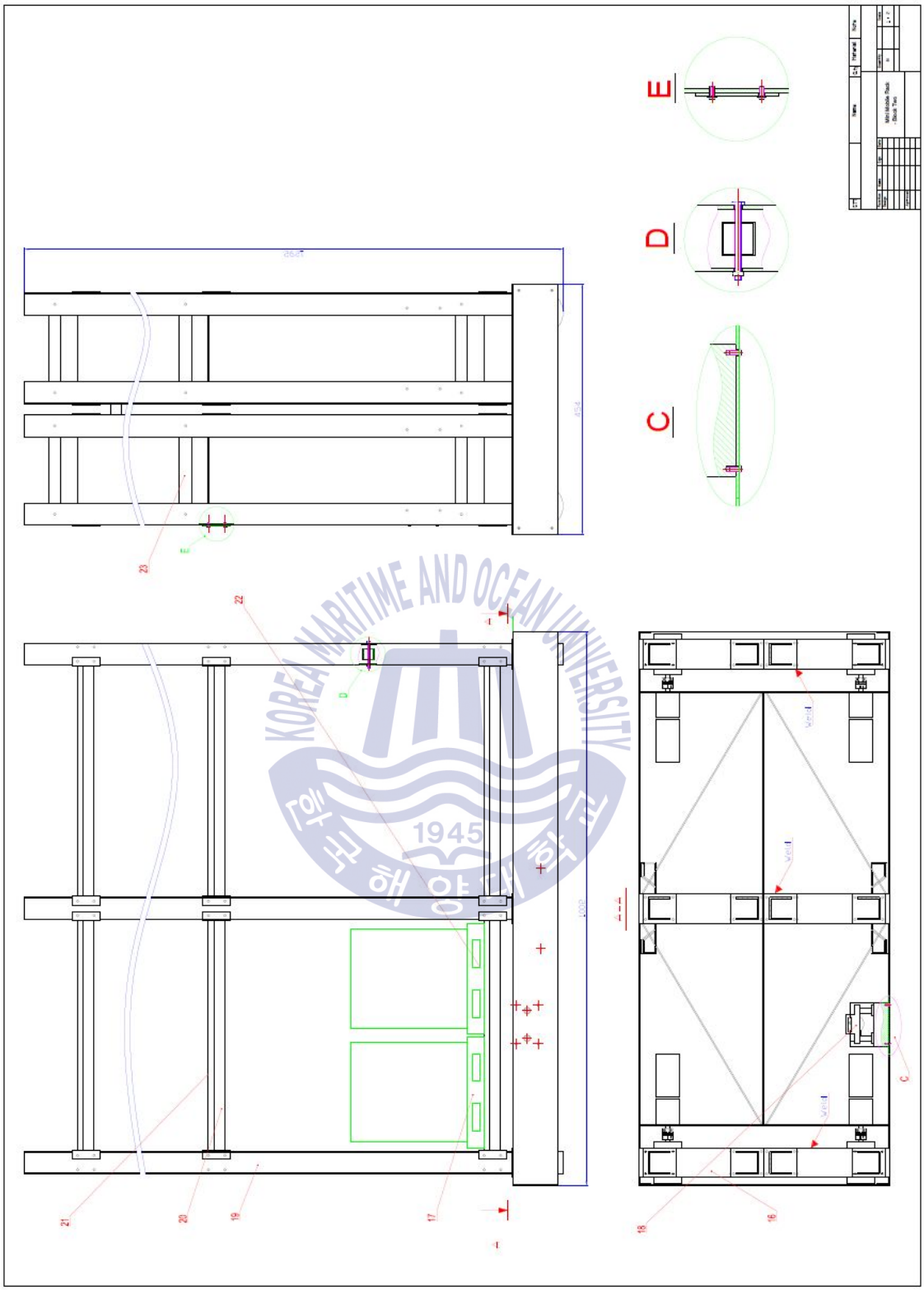

그림 4.4 테스트 모듈의 도면 2

Fig. 4.4 Drawing of test module 2

이동랙은 베이스 모듈과 상부 프레임으로 이루어져 있으며 세부적인 내용은 다음 3D 도면을 통해 확인한다.

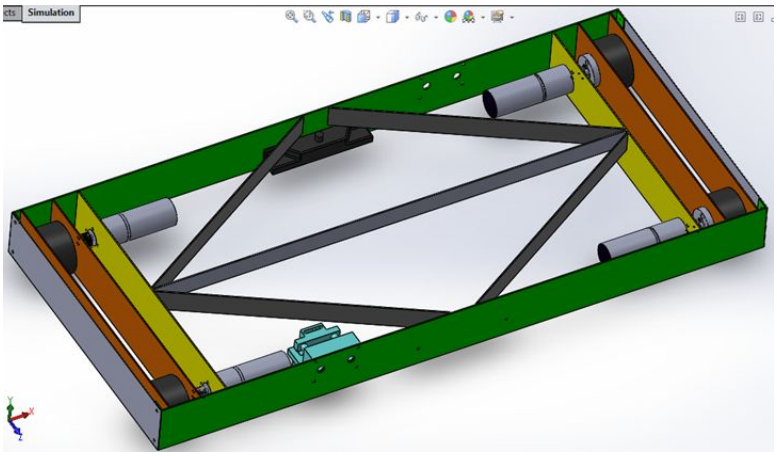

그림 4.5 베이스 모듈의 3D 도면
Fig. 4.5 3D drawing of base module

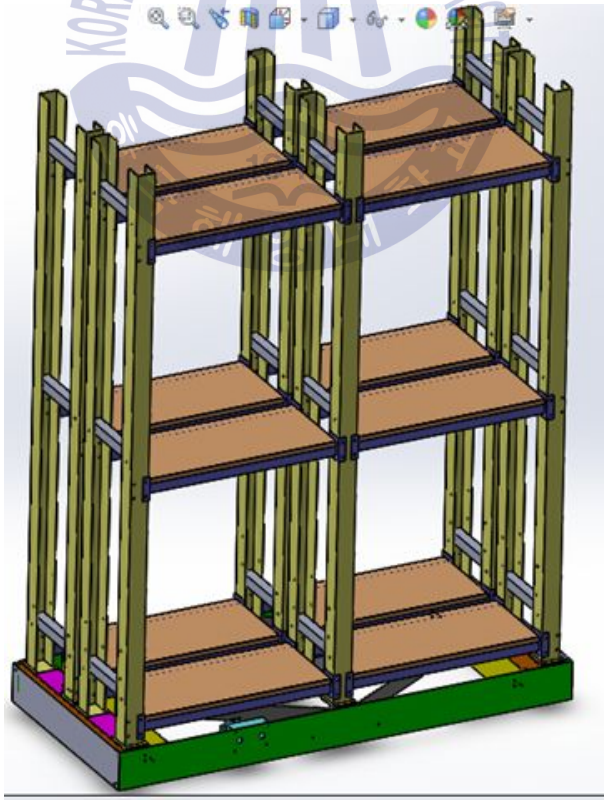

그림 4.6 이동랙의 3D 도면
Fig. 4.6 3D drawing of mobile rack

베이스 모듈은 구동부와 컨트롤 파트가 장착 되어있으며 좌우, 앞뒤로 4개의 모터에 휠이 장착되어 작동되고 상부 프레임은 본 연구의 초기부터 고려된 3층의 형태로 되어져 있다.

베이스 모듈 위에는 상부의 프레임이 올라가게 되고 이를 지탱해야 하므로 모터와 휠 부분은 직접 연결시키지 않는다. 샤프트, 베어링 등 여러 동력 전달 파트로 이루어져 위쪽에서부터 작용하는 이동랙의 전체적인 무게를 지탱한다.

이처럼 설계된 이동랙 모델의 완성품은 아래와 같다.

그림 4.7 이동랙 모델

Fig. 4.7 Model of mobile rack

4.3 실험결과 및 고찰

4.3.1 실험 조건

실험은 구동부의 개별 테스트 및 전체 조립 후 테스트로 이루어진다. 전체 조립 테스트의 경우 무부하 상태에서 한쪽에만 약간의 부하를 준 상태로 진행하였으며 이동거리는 일정하게 반복적인 움직임을 적용하였다.

4.3.2 실험 결과 및 고찰

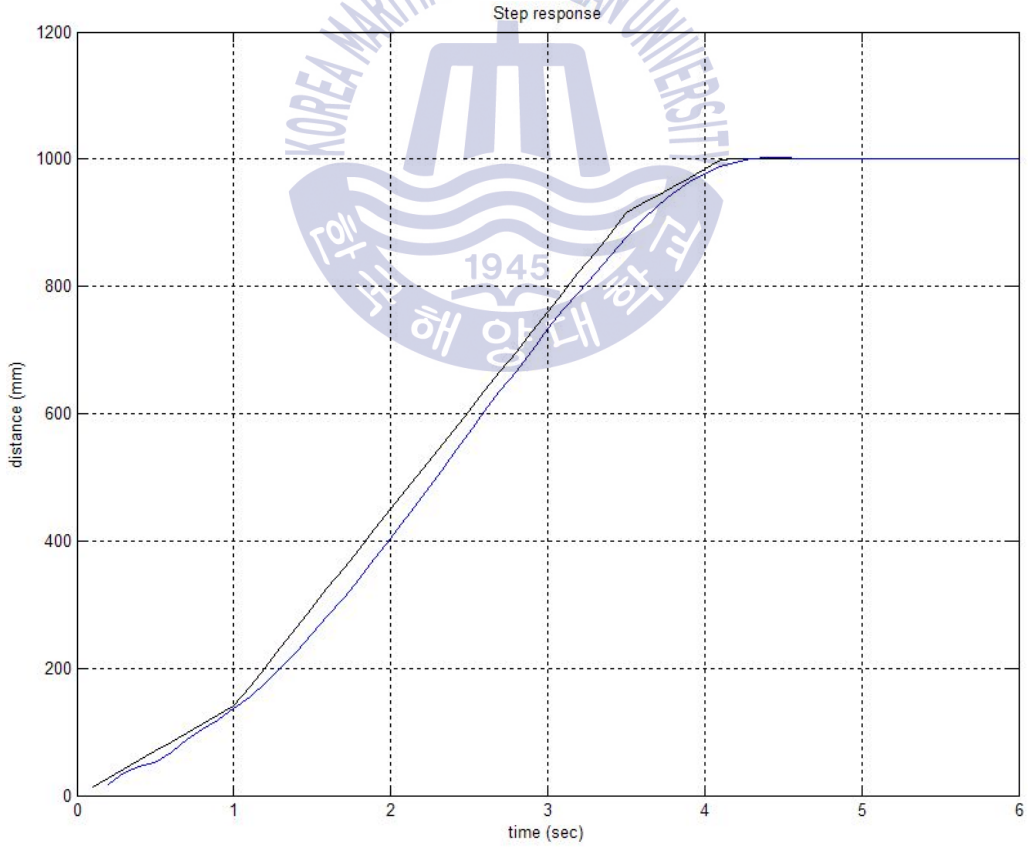

그림 4.8 서보 제어가 적용된 모터의 기본 스텝응답

Fig. 4.8 The basic step response of the motor servo control is applied

앞의 그래프 그림 4.8은 컨트롤러로부터 모터 드라이버로 입력이 인가되었을 때의 엔코더 데이터이다. 실제 모델의 구동을 할 때에도 가속과 감속은 최대의 가감속을 하는 것이 아닌 천천히 가속하고 멈출 때에도 천천히 감속해야한다.

아래의 그래프 그림 4.9은 구동 모터 두 개를 이용해 반복제어 및 동기화제어를 적용하기 전에 두 모터 사이의 오차를 측정한 결과이다.

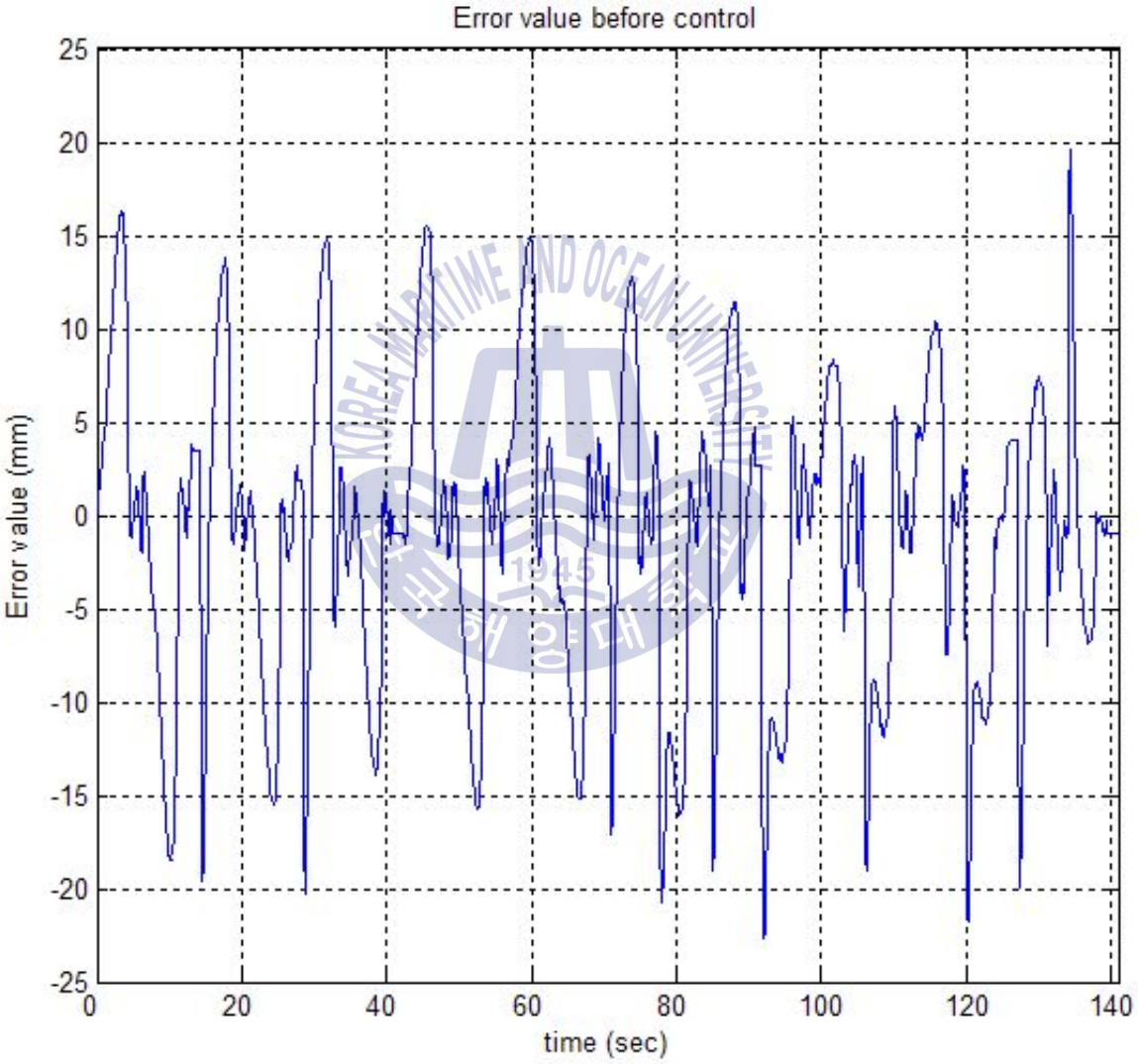

그림 4.9 두 개의 구동부를 이용하여 측정한 제어 전 오차값

Fig. 4.9 Error value before control was determined using the two drive parts

다음 그래프인 그림 4.10은 반복적으로 동기화 제어를 적용하여 주기별로 오차를 개선한 테스트 결과로써 앞서 시뮬레이션 한 것처럼 초기 오차는 비슷하지만 전체적으로 오차가 많이 줄었으며 주기를 반복해가며 반복적으로 구동을 할수록 마지막에는 오차가 줄어든 것을 확인할 수 있다.

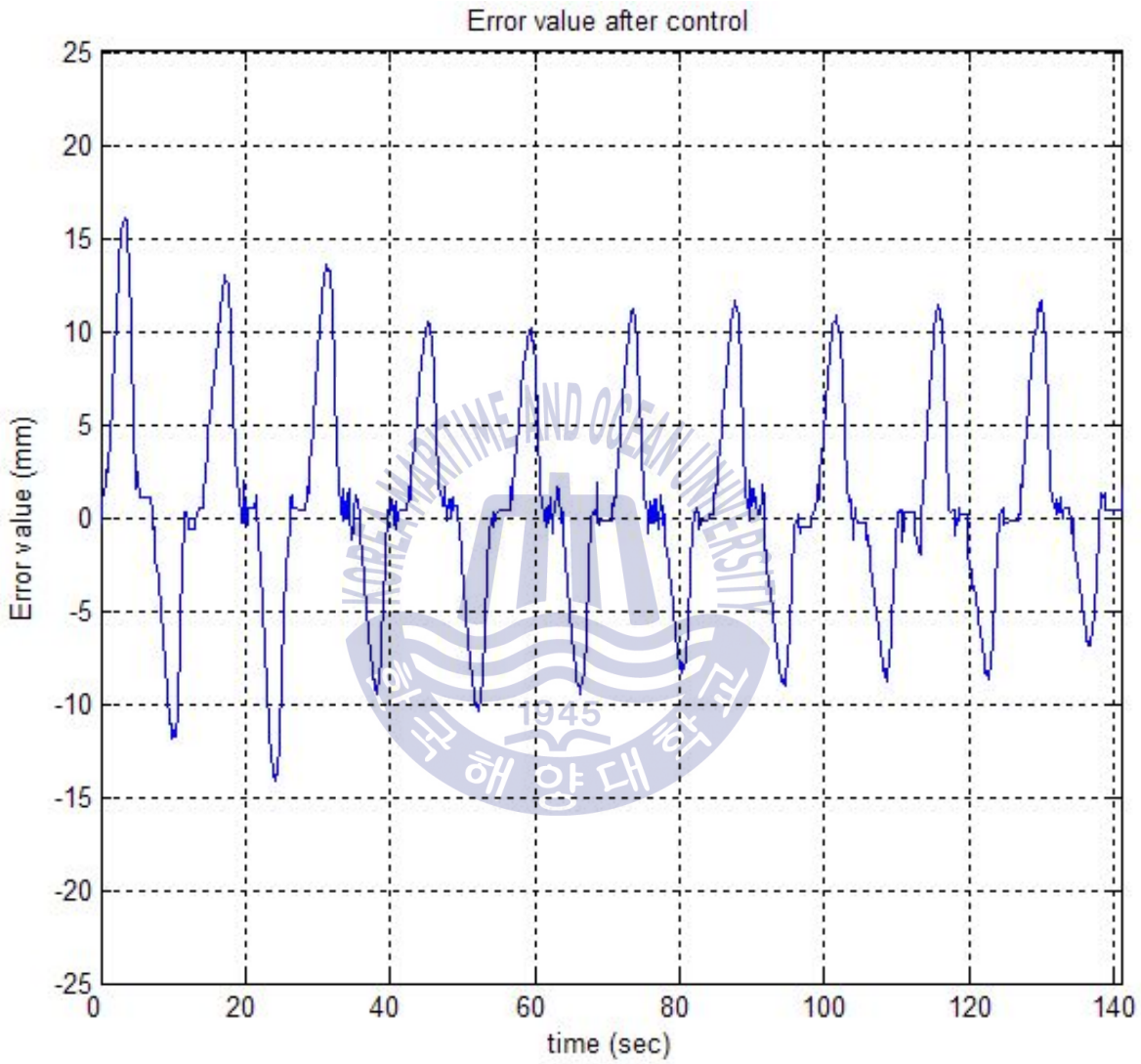

그림 4.10 두 개의 구동부를 이용하여 측정한 제어 후 오차값

Fig. 4.10 Error value after control was determined using the two drive parts

마지막으로 모델링부터 설계, 시뮬레이션 그리고 모듈별 테스트까지 진행하면서 확인한 반복제어계를 통한 동기화 제어를 실제 이동랙과 유사하게 제작한

이동랙 모델에 적용해보았다.

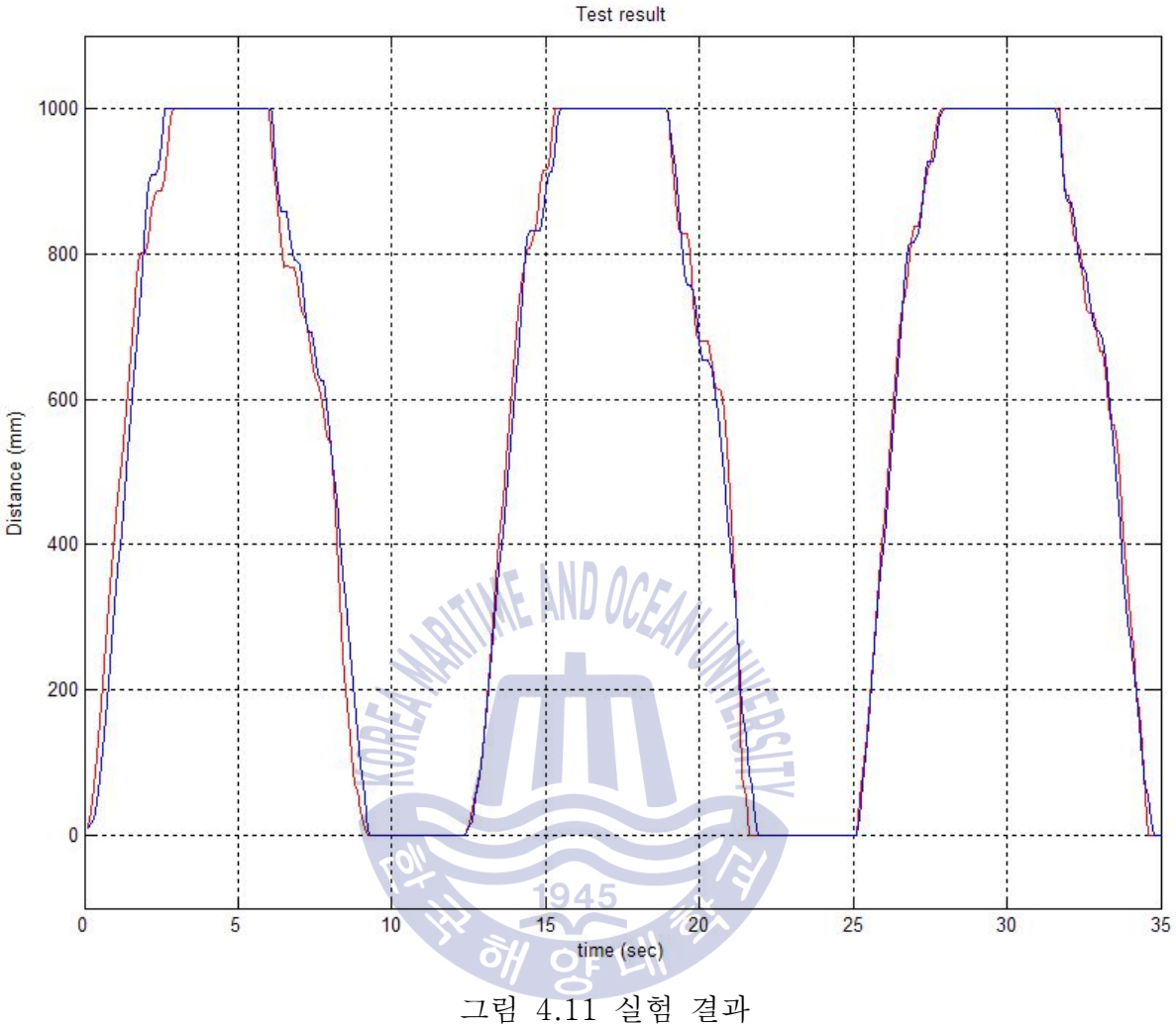

그림 4.11 실험 결과

Fig. 4.11 Test result

실험 결과는 위의 그림 4.11 같이 나타났다. 초반 주기에서의 이동시에는 약간의 오차를 가진 상태로 이동을 한다. 하지만 반복적인 제어와 동기화 제어가 진행되면서 이동할 때의 오차들은 크게 줄어들어 동일하게 움직이는 것을 확인할 수 있었다.

아래 그림 4.12는 그림 4.11 실험 결과의 오차 값을 표현한 그래프로 확인을 해보면 처음 주기에는 오차가 크게 나타나지만 반복제어를 통해 주기가 반복되면서 오차가 점점 줄어드는 것을 확인할 수 있었다.

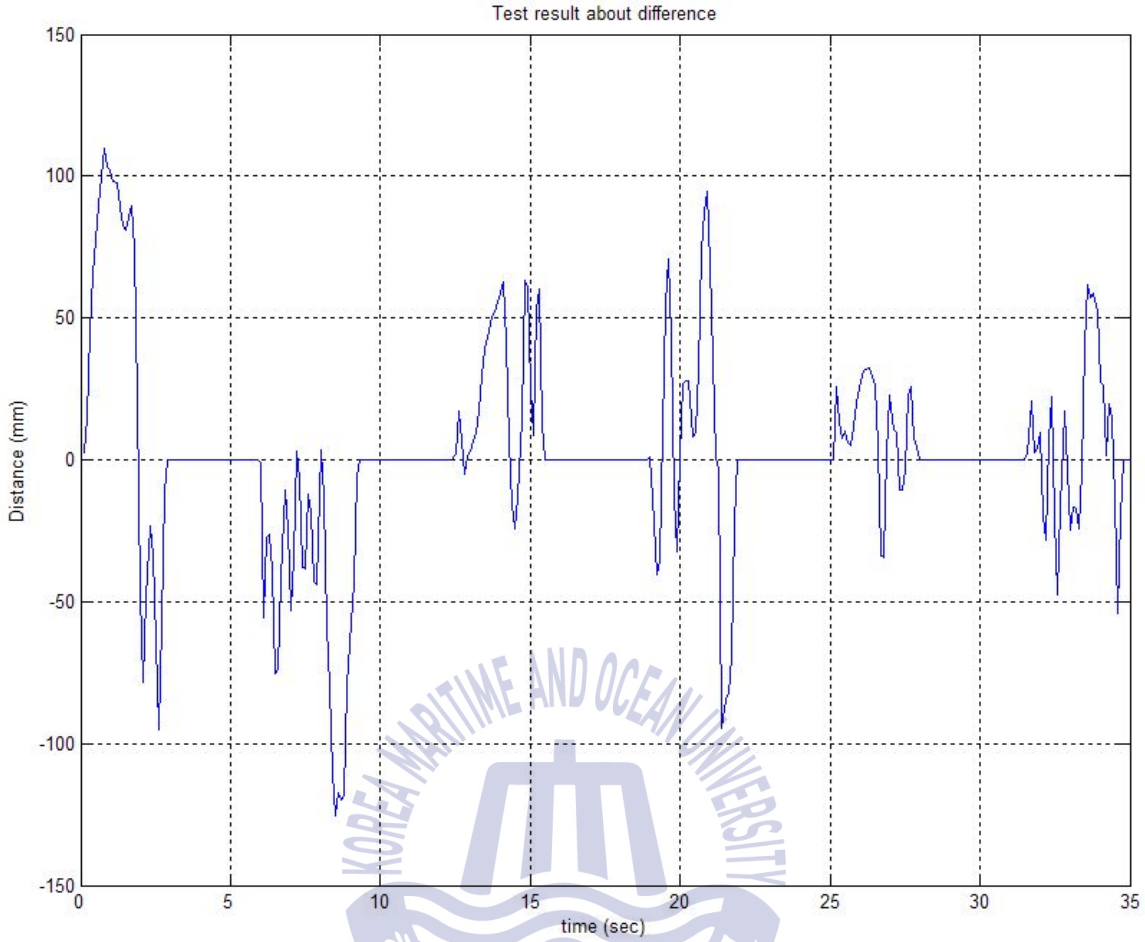

그림 4.12 실험 결과의 오차값

Fig. 4.12 Error value of test result

제 5 장 결 론

이번 연구 논문에서는 무궤도 이동랙이라는 시스템에 내부적으로는 구동휠 간의 동기화제어 그리고 외부적으로는 각 이동랙간의 동기화제어를 행하는 2중형 반복제어계를 제안했다. 그에 앞서 각 휠에는 고중량이라는 파라미터를 고려하여 다양한 화물 변동을 특징으로 가지므로 이러한 파라미터 변화에 대응할 서보제어기를 설계한다.

실제 이동랙을 통해 수학적 모델링을 하여 특성을 파악하고 이를 이용하여 시뮬레이션을 진행하였으며 시뮬레이션에는 각 구동부에는 서보 제어기를 적용하고 구동휠 간의 동기화 제어와 다중으로 이루어진 이동랙 간의 동기화 제어를 반복적으로 적용하여 MATLAB을 이용해 분석하였다.

시뮬레이션을 통해 확인된 반복제어계를 이용한 동기화 제어계를 실제 이동랙에는 적용하여 테스트하기 어려우므로 이를 테스트 하기 위한 테스트 모듈을 구상하였으며 간단하게 제어부, 구동부 및 센서부로 이루어지도록 설계 제작

우선 구동 모듈별로 알고리즘을 적용하여 테스트 하였고 시뮬레이션 결과와 비교하여 시뮬레이션한 제어 알고리즘이 잘 적용되었는지 확인하였다. 이어서 베이스 모듈 및 상부 프레임이 모두 구성되어 있는 테스트 모듈에 본 연구 내용을 테스트하였다.

그 결과 모델링 및 시뮬레이션 과정에서 확인하지 못했던 오차 요인들이 발견되었는데 이러한 오차들은 동력전달부 부품의 재료 특성과 각 파트별 전력 및 신호 간섭 등 고려하지 못했던 부분들의 오차로써 추후 연구에는 이를 고려하고 더욱 세부적으로 모델링하고 제어계를 설계하여 보완해야할 것으로 분석된다.

참 고 문 헌

[1] Katsuhiko Ogata, *Modern control engineeringn fourth edtion*, (주)사이텍미디어, 2003.

[2] 김상봉, 하주식, *기초 시스템 이론*, 한미, 1990.

[3] 古田勝久, 川路茂保, 美多 勉, 原 辰次, メカニカルシステム制御, オーム社, 1984

[3] Raymond A. Serway, Jerry S. Faughn, Chris Vuille, *College Physics, 8th Edition*, CENGAGE Learing, 북스힐, 2011.

[4] 金相奉, 河注植, Mechanical system control, *한미*, 1992

[5] 정슬, 제어시스템 분석과 MATLAB 및 Simulink의 활용, *청문각*, 2010

[6] Palm, William J., 공학도를 위한 매트랩7, *교보문고*, 2009

[7] 허경용, 아두이노 상상을 스케치하다, *제이펍*, 2014

[8] Michael Margolis, 윤순백, 레시피로 배우는 아두이노 쿡북, 제이펍, 2014

[9] Sang-Bong Kim, Soo-Heung Park, JIn-Ho Suh, In-Kyu Kim, and Hwan-Seong Kim, “NOx Reduction Control for SCR System Using Repetitive Control Method in a Refuse Incineration Plant”, 1998

[10] Hwan-Seong Kim, Kawaji Shigeyasu, “A study on the modelling and its characteristic of AS/RS stacker crane in high rack automated warehouse system”, 1999.