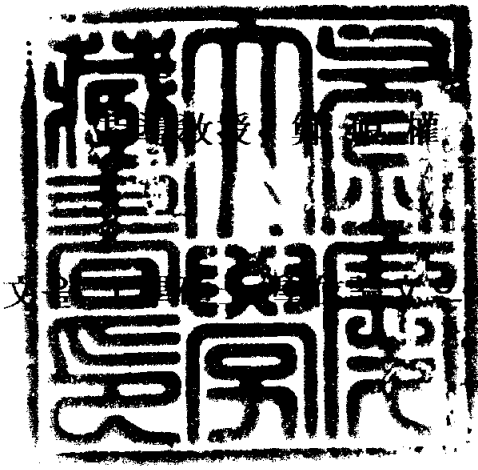
工學碩士 學位論文

외란에 강인한 다축 시스템의 정밀 위치동기제어에 관한 연구



이 論文을 [illegible] 提出함

2004년 2월

釜慶大學校 大學院

메카트로닉스工學 協同課程

崔 峯 碩

崔峯碩의 工學碩士 學位論文을 認准함

2003年 12月 26日

主　審　工學博士　梁 注 鎬　(印)

委　員　工學博士　張 志 城　(印)

委　員　工學博士　鄭 碩 權　(印)

목 차

A Study on Precise Position Synchronous Control of Multi-Axes System with Robustness against Disturbance

Bong-Seok Choi

Department of Mechatronics Engineering, Graduate School
Pukyong National University

Abstract

The position synchronous control of multi-axes servo system is considered to be an important subject in motion control field for application to plotters, robot arms, and numerical control machines. In the past, position synchronous controls are mainly realized with mechanical coupling such as gears, cams, and shafts etc. However recently, owing to the development of mechatronics technology, position synchronous is realized with software algorithms without complicated mechanical couplings. Comparing to the conventional methods, newly devèloped method has some merits such as good flexibility, low cost for maintenance, and decrease in environmental problems such as noises and vibrations.

In this paper, we deal with a precise position synchronous control of four-axes system which is working under various load disturbances. Each axis driving system consists of a speed controller and an acceleration controller as an inner loop instead of conventional current control scheme. The acceleration control plays an important roll to suppress load disturbances quickly. Also, each axis is coupled by a maximum position synchronous error comparison to minimize position synchronous errors according to integration of speed differency. As a result, the proposed system enables precise

synchronous control with good robustness against load disturbances during transient as well as steady state. The stability and robustness of the proposed system are investigated through its frequency characteristic and numerical simulations. Finally, experimental results under load disturbances demonstrate the effectiveness of the proposed control system for four-axes position synchronous control.

제 1 장 서 론

1.1 연구배경

다축 위치동기 서보시스템은 메카트로닉스 기술(mechatronics technology)의 활용 분야로서, 주행 대차구동시스템, 플로터, 각종 수치제어 공작기계 및 FMS(Flexible Manufacturing System), CIM(Computer Integrated Manufacturing) 시스템 등과 같은 자동화 분야에서 많이 활용되어지고 있는데, 이런 시스템의 정밀한 위치동기제어는 제품의 품질 향상 및 제조공정 및 작업효율에 중요한 영향을 미친다.

일반적으로 종래의 다축 시스템의 위치동기는 주로 거대한 대용량의 단일 모터의 축에 복수의 기어(gear)와 캠(cam)등을 결합하여 기계적으로 연동시키는 집중구동 방식의 위치동기제어가 대부분이었다. 이 기계적인 커플링에 의한 위치동기방식은 동작의 전달이 정확하고, 기계적인 결합만으로 간단히 위치동기를 취할 수 있는 장점은 있다. 하지만 장기운전으로 인한 기계요소의 마모발생, 장치의 거대화, 작업가동률 저하, 에너지 이용 면에서의 불리함, 마찰 및 소음·진동 발생 등과 같은 결점을 다수 가지고 있다.

최근, 기존의 기계적인 커플링에 의한 집중구동 방식의 위치동기에서 벗어나 메카트로닉스 기술을 활용한 분산구동 방식이 제안되었다[1~9]. 이 방식은 기존의 하드웨어적인 방식에서 탈피하여 소프트웨어방식으로, 바꾸어 말하면 물리적 커플링에 의한 방식에서 제어 알고리즘을 이용하여 복수의 축을 독립된 모터로 개별 구동하는 방식으로 위치동기를 취하는 것이다. 이러한 분산구동 방식의 장점은 기존의 방식들의 결점을 대폭 보완할 수 있으나, 복수의 축 간의 정밀한 위치동기를 위해서는 적절한 위치동기제어가 필수적으로 요구되어진다.

일반적으로 연속 회전하는 다축 시스템의 위치동기제어는 각 축이 플랜트와 제어기들로 구성되며, 고정도의 정밀 위치동기를 위해서는 각 축

에 다양한 외란이 인가되는 상황 하에서 정상상태 뿐만 아니라 과도상태에서도 엄밀한 위치동기가 유지되도록 제어계를 설계해야만 한다. 그러나, 각 축이 서로 다른 동특성을 가지므로, 전 운전영역에서 정밀한 속도 및 위치동기를 얻는 것은 용이하지 않다.

이러한 배경으로 위치동기제어에 관한 다양한 연구가 진행되어져 오고 있지만, 대다수의 연구가 다음과 같은 문제점을 내포하고 있다.

1) 구동축의 축 수에 있어서 단지 2축간의 위치동기에 중점을 둔 2축 시스템의 위치동기제어에 국한된 연구가 대부분이었다[1~5].

2) 위치동기제어의 방식으로 마스터-슬레이브(Master-Slave, 이하 M/S방식)방식은 슬레이브측에 외란이 인가된 경우는 적절히 대응하기 어렵고, 마스터측에 외란이 인가된 경우 슬레이브측의 속도희생이 필수적이므로, 빠른 보상을 요하는 정밀 위치동기시스템에는 적절한 보완책이 필요하다[1]. 또한, 2축간의 협조제어(Cooperative Control)방식은 시스템이 3축 이상으로 확장될 경우, 최선의 위치동기를 취하기 위해서는 그 적용이 그리 간단하지 않다[2].

3) 각 축의 제어계의 내부루프로서 전류제어계를 갖는 경우는 수시로 변동하는 외란이나 각종 마찰성분에 대해 적극적으로 대처하기 힘들며, 위치동기제어 실험에 있어서 외란 인가하의 실험은 적극적으로 고려되지 않았고, 주로 시뮬레이션에 국한된 연구가 대다수였다[7].

4) H_∞제어와 PID 제어 등을 혼합한 다양한 제어기 설계법이 제시되었지만, 제어기의 게인설정이 복잡하고 주로 시행착오인 반복법에 의존하고 있어 설계과정 등이 번거롭다[3~4].

5) 다축으로의 확장성을 고려한 연구로서 기준 모델과 실제 모터 축과의

위치오차를 검출한 후 이를 영으로 수렴시키는 방법이 제안되었지만[3], 기준 모델의 작성 자체가 각 축에 사용된 모터나 부하측의 동특성이 상호 유사할 경우에 가능한 것으로, 전기적 및 기계적 시정수가 각기 다른 모터와 부하들로 연결된 일반적인 다축 시스템에는 적용이 용이하지 않다.

1.2 연구목적 및 내용

위치동기시스템의 성능은 그 특성상 최대 위치동기 오차의 크기에 전적으로 좌우된다. 또한, 최대 위치동기오차는 기본적으로 외란 인가시에 발생되며 각 축들 간에 작용하는 외란이 역상일 경우에 그 크기가 가장 크다. 따라서, 다축 시스템의 정밀한 위치동기를 위해서는 우선 각 축이 외란에 대해 충분한 강성을 가질 것과 빠른 속도 추종성을 가져야 한다. 우수한 속도제어계의 설계에도 불구하고, 실제 시스템은 각 축이 갖는 동특성의 차로 인해 지령치와 응답치 사이에 미소한 속도 차가 발생하며 각 모터에 발생되는 각기 다른 속도 차들의 적분치가 축 간의 위치동기 오차로서 나타난다. 따라서 이 위치동기오차를 신속히 제거하는 것이 고정도 위치동기의 주요한 관건이 된다. 이때, 특정한 축만의 속도를 제어하는 것 보다, 각 축의 속도 희생을 최소화하는 방향으로 모든 축이 상호 협조하여 부하를 분담하는 방법이 위치동기오차의 신속한 제거에 훨씬 효과적이다.

본 논문에서는 이상의 점을 감안하여 가속도제어[11]와 최대 위치동기오차 비교법에 의한 4축 시스템의 정밀한 위치동기제어 방식을 제안한다. 제안하는 방식의 특징은 빠른 외란 제거를 위해 기존의 전류제어기 대신 가속도제어기를 갖는다는 점이다. 가속도제어기는 기존의 전류제어기에 비해 우수한 외란 제거 성능을 가지므로 정상상태의 경우에서는, 계단상의 외란 인가시 위치동기제어기 없이도 위치동기오차를 신속히 영으로 수렴시킨다. 그리고 앞서 기술한 바와 같이, 위치동기제어계는 다양한 외란 하에서 과도상태를 포함한 전 운전 영역에 걸쳐 정밀한 위치동기가

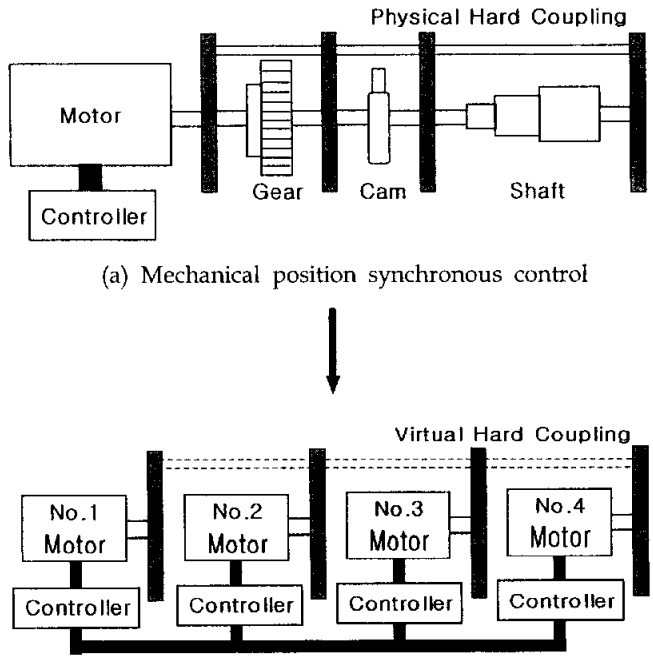
요구되므로 이를 고려하여 위치동기제어기를 설계한다.

기존의 2축 위치동기시스템에 적용된 협조제어형 위치동기제어 방식은 본 연구에서 상정하는 4축 시스템의 경우에는 그 적용이 어렵다. 따라서, 본 연구에서는 협조제어방식을 수정한 최대 위치동기오차 비교법을 제안한다. 이 방식은 각 축의 위치 정보를 나머지 축들과 축차 비교한 후, 최대의 오차를 갖는 상대 축에 주목하여 이 두 축간의 위치동기오차를 우선적으로 보상하는 방식이다. 이 방식의 특징은 각 축의 속도 희생을 최소화하면서 빠르게 위치동기를 취할 수 있다는 점이다.

각 제어기의 설계시, 현장에서 가장 보편적으로 사용되고 있는 PI(Proportional, Integral) 제어칙을 이용하였으며, 속도 및 가속도제어기는 PI제어기를 갖도록, 그리고 위치동기제어기는 전체 시스템 설계의 편의성과 속응성을 동시에 고려하여 단순 P제어기로만 설계하였다. 또한, 각 제어기의 게인 결정은 시행착오를 최소화할 수 있도록 제안하였다. 설계된 가속도제어계에 대해서는 안정성 검토를 행하였으며, 실제 모터 모델을 반영한 외란인가 하에서의 다양한 수치 시뮬레이션 및 외란 인가 장치를 갖는 4축 위치동기제어 시스템을 구성한 후, 실기 실험을 통해 제안한 방식의 이론 및 유효성을 최종적으로 확인하였다.

제 2 장 다축 구동계의 위치동기제어

Fig. 2.1에 다축 구동계의 기본적인 개념도를 보인다. 기어나 캠, 샤프트 등과 같은 기계요소들을 다수 결합하여 기계적으로 위치동기를 취하던 Fig. 2.1(a)와 같은 방식에서 벗어나, 최근 소프트웨어적인 제어 알고리즘을 이용하여 위치동기를 실현하는 Fig. 2.1(b)와 같은 방식이 대두되었다. 본 논문에서는 다축 위치동기제어계의 설계에 있어 우선, 각 축의 속도제어계의 설계를 보이고, 최대 위치동기오차 비교법에 근거한 4축 위치동기제어계의 설계를 보인다.



(a) Mechanical position synchronous control

(b) Position synchronous control by software algorithm

Fig. 2.1 Basic concept of proposed control system

2.1 모터 구동계의 모델링

구동계의 동특성은 DC 모터의 전기계에 대한 키르히호프의 회로방정식과 기계계의 회전체에 대한 뉴우턴의 운동방정식을 적용하면 일반적으로 다음과 같이 기술된다.

$$L_a \frac{d}{dt} i_a(t) + R_a i_a(t) = v_a(t) - K_e \omega(t) \tag{2.1}$$

$$J \frac{d}{dt} \omega(t) + b\omega(t) = K_t i_a(t) - T_l \tag{2.2}$$

여기서, 각 파라미터의 명칭은 아래와 같다.

$v_a(t)$	: 전기자 전압	$\omega(t)$	: 기계 각속도
$i_a(t)$	: 전기자 전류	R_a	: 전기자 저항
L_a	: 전기자 인덕턴스	J	: 회전체 이너셔
b	: 마찰계수	K_t	: 토크정수
K_e	: 유기전압 정수	T_l	: 부하외란 토크

식 (2.1)과 식 (2.2)를 라플라스 변환시키고, 그 초기치들을 0으로 두면, 다음의 식 (2.3)과 식 (2.4)로 표현된다.

$$(L_a s + R_a) i_a(s) = v_a(s) - K_e \omega(s) \tag{2.3}$$

$$(Js + b) \omega(s) = K_t i_a(s) - T_l(s) \tag{2.4}$$

Fig. 2.2는 식 (2.3)과 식 (2.4)를 이용한 모터 모델의 블록선도를 보인다. 그리고 이하의 제어기 설계시, 역기전력 K_e는 설계의 편의상 제외하였다.

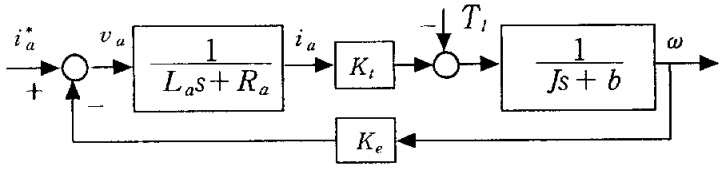


Fig. 2.2 Block diagram of motor modeling

2.2 단일축 속도제어계의 설계

일반적으로 다축 위치동기시스템은 각 축이 속도지령에 충실히 추종하는 속도제어계가 설계되어야 한다. 본 연구에서 제안하는 속도제어계는 기존의 전류제어계 대신에 부하외란 및 각종 마찰토크에 강인한 가속도 제어기와 속도지령 추종을 위한 속도제어기로 구성된다.

2.2.1 가속도제어기의 설계

서보계는 목표치에 대한 응답 특성뿐만 아니라 외란 제거 성능이 매우 중요하다. 일반적으로 전기모터를 이용한 서보계는 모터 전류를 제어하는 내부루프를 통해 외란에 대한 빠른 보상을 행하고 있다. 그러나, 시스템을 운동제어(motion control)의 관점에서 보면, 위치의 미분은 속도이고 속도의 미분은 가속도이므로, 엄밀히 모터 전류나 토크는 아니다. 그리고 시스템 전체의 가속도는 부하외란 및 각종 마찰토크를 포함한 총합적인 구동력에 의해 정해지므로 가속도를 직접적으로 제어할 수 있다면 자동적으로 강인한 외란 제거 성능을 확보할 수 있다. 그러나, 적절한 가속도 센서의 부재 및 전류의 최대치 억제 필요성 등으로 인해 현재까지의 서보계는 내부루프로서 거의 전류제어계를 이용하였다. 그러나, 최근에는 다양한 가속도 센서가 개발되고 있으며, 관측기 이론 등을 통해 가속도 추정 알고리즘도 많이 개발되어 가속도를 직접 제어하려는 시도가 활발해 지고 있다.

내부에 전류제어계를 갖는 구조는 서서히 변화하는 외란에 대해서는 무한대에 가까운 강성을 가지지만, 빠르게 변화하는 외란에 대해서는 동적강성이 구동계의 전체 J 값에 크게 의존하게 되는 문제점을 갖는다. 그러므로 가속도를 연산하여 보상할 수 있다면 실제로 동적강성을 늘리는 것과 동일한 효과를 얻을 수 있어, 빠르게 변하는 외란토크에 대해서도 강인한 서보계를 구축할 수 있다. 이하에서는 모터의 속도제어기 설계시 널리 사용되고 있는 PI설계기법을 이용한 가속도제어기 G_a 및 속도제어기 G_s의 설계 과정을 보인다.

가속도제어계의 입력인 가속도지령에서 출력인 가속도 출력까지의 블록도는 Fig. 2.3과 같이 나타내어진다.

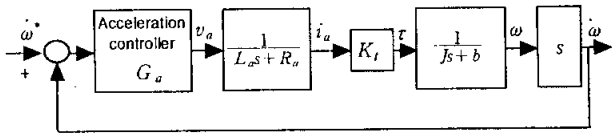


Fig. 2.3 Block diagram of acceleration control system

가속도제어기 G_a의 출력인 전압지령 v_a는 PI 제어칙을 이용하여 식 (2.5)와 같이 얻어진다.

$$v_a = G_a(\dot{\omega}^* - \dot{\omega}) = K_{ap}\left(1 + \frac{1}{T_{ai}s}\right)(\dot{\omega}^* - \dot{\omega}) \tag{2.5}$$

여기서, K_{ap}와 T_{ai}는 설계해야할 가속도제어기의 비례게인과 적분시간을, 위 첨자 '*'는 지령치를, '·'는 시간미분을 각각 나타낸다.

피드백제어에 의해 필요한 설계사양이 충족된다면 개루프 전달함수는 간단한 형일수록 좋으므로 제어기의 적분시간을 모터 전기계의 시정수로 식 (2.6)과 같이 정한다.

$$T_{ai} = \frac{L_a}{R_a} \tag{2.6}$$

이때, 마찰계수 b를 서보게임을 감안하여 0으로 간주할 경우, 가속도제어계의 개루프 전달함수 및 폐루프 전달함수는 식 (2.7)과 식 (2.8)로 간략화 된다.

$$G_a{}^0 = \frac{1}{Ks} \tag{2.7}$$

$$G_a{}^c = \frac{1}{(Ks+1)} \tag{2.8}$$

여기서, $K = L_a J / K_{ap} K_t$ 이다.

결국, 적분시간을 모터 전기계의 시정수로 설정함에 따라, 개루프 전달함수는 단순 적분요소로, 폐루프 전달함수는 1차 지연요소로 나타난다. 이때 개루프 및 폐루프 전달함수의 주파수 특성은 $\omega_c = K_{ap}K_t / L_a J$ 의 교차각주파수에서 -45[°]의 위상각을 가지며 정상상태에서 0[dB]의 값을 갖는 안정한 특성을 보인다. 이것은, 정상상태에서는 가속도제어계의 가속도 지령이 그대로 가속도제어계의 출력으로 되는 것을 의미하는 것으로, 결국 속도제어계의 설계시 가속도제어계를 무시한 속도제어계의 설계가 가능하게 되는 것을 나타낸다.

교차각주파수에서 K_{ap}를 제외한 나머지 파라미터는 모터의 사양에 의해 결정되는 정수이다. 따라서 비례게인 K_{ap}를 크게 할수록 주파수 대역폭이 증가하므로 속응성이 개선될 수 있음을 알 수 있다. 그러나, 실제적으로 이 값은 디지털계에서의 구현일 경우 프로그램의 샘플링주파수의 설정 및 A/D, D/A 변환기와 검출기 등의 성능의 영향에 의해 이론치보다 작은 값으로 제한되어야 한다. 통상 선형시스템에서의 실현을 고려한다면 가속도제어계의 교차각주파수 ω_c는 PWM 케리어주파수 f_{car}와 관련하여 $\omega_c \leq K_d$와 같은 제약을 받게 되며, 이산계에서 대략 K_d 값은 케리어주파수의 $2\pi/3$의 값을 갖는다. 이 관계로부터 가속도제어기의 비례게인은 식 (2.9)와 같이 결정할 수 있다.

$$K_{ap} = \frac{K_d L_a J}{K_t} \tag{2.9}$$

여기서, $K_d = \frac{2\pi}{3} f_{car}$ 이다.

2.2.2 속도제어기의 설계

다축 시스템의 정밀 위치동기를 위해서는 우선 각 축이 오버슈트(overshoot)를 최소화하면서, 동시에 짧은 정착시간(settling time)을 가지며 지령속도에 추종해야한다. 가령 속도응답에서 오버슈트나 긴 정착시간을 갖는다면 과도상태나 외란인가 시, 각 축의 동특성으로 인하여 빠른 위치동기는 더욱 어렵게 된다. 따라서 전 운전영역에서 속도지령과 응답의 차인 속도오차를 최소화시키는 속도제어기를 설계할 필요가 있다.

Fig. 2.4는 가속도제어계를 포함한 속도제어계의 블록도를 나타낸다.

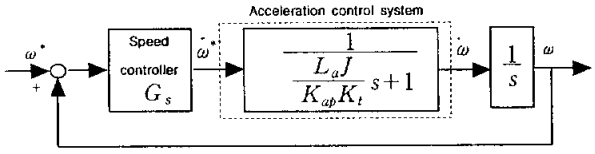


Fig. 2.4 Block diagram of speed control system

PI 제어칙에 의한 속도제어기의 전달함수 $G_s(s)$는 식 (2.10)으로 주어진다.

$$G_s(s) = K_{sp}(1 + \frac{1}{T_{si}s}) = (K_{sp} + \frac{K_{si}}{s}) \tag{2.10}$$

여기서, T_{si}는 적분시간, K_{sp}와 K_{si}는 비례게인 및 적분게인이다. Fig. 2.4에 있어, 속도지령에서 속도출력까지 개루프 전달함수 $G_s^o(s)$는 식 (2.11)과 같이 유도되며, 그 주파수 특성은 Fig. 2.5와 같다.

$$G_s^o(s) = (K_{sp} + \frac{K_{si}}{s}) \cdot \frac{1}{\frac{L_a J}{K_{ap} K_t} s + 1} \cdot \frac{1}{s} \tag{2.11}$$

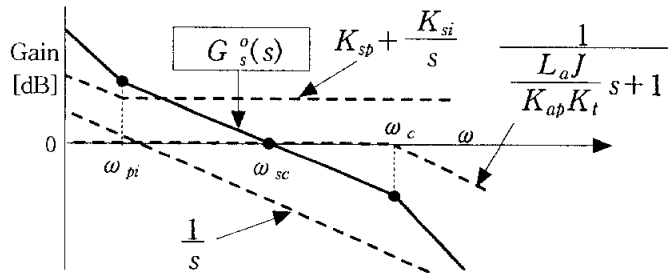


Fig. 2.5 Open loop frequency characteristic of speed control system

Fig. 2.5의 실선은 식 (2.11)의 세 요소들에 대한 각 주파수함수의 게인 선도(점선 표시)를 모두 합한 것이다. 여기서, 속도제어계의 교차각주파수 ω_{sc}가 가속도제어계의 교차각주파수 ω_c의 $1/m_1$배 이하로 되면 가속도제어계는 ω_{sc}부근에서 $G_a^c(s)\cong 1$로 근사화 된다. 한편, 속도제어기의 절점각주파수 ω_{pi}는 $\omega_{pi}=K_{si}/K_{sp}$이므로, 이 ω_{pi}가 ω_{sc}의 $1/m_2$ 배 이하인 경우 ω_{sc}의 부근에서는 $G_s(s)\approx K_{sp}$와 같이 근사화 된다.

이상의 가정들이 근사적으로 성립되는 영역에서 PI 속도제어계의 개루프 전달함수는 각주파수 ω가 교차각주파수 ω_{sc}의 부근에서 식 (2.12)로 근사화 된다.

$$G_s^o(s)\fallingdotseq K_{sp}\cdot\frac{K_t}{Js} \tag{2.12}$$

여기서, $|G_s^o(j\omega_{sc})|=1$이 되도록 하는 비례게인 K_{sp}는 식 (2.13)과 같이 구해진다.

$$K_{sp}=J\cdot\frac{\omega_{sc}}{K_t}=\frac{J\omega_c}{K_t m_1} \tag{2.13}$$

한편, $\omega_{pi}=K_{si}/K_{sp}$ 이고, 또한 $\omega_{pi}=\omega_c/(m_1m_2)$이므로, 이들 관계를 이

용하면 적분게인 K_{si}는 식 (2.14)와 같이 구해진다.

$$K_{si} = K_{sp} \cdot \frac{\omega_c}{m_1 m_2} \tag{2.14}$$

식 (2.13) 및 식 (2.14)에서 m_i를 제외한 나머지 파라미터는 이미 알고 있는 값이다. 따라서 적절한 m_i를 선정하면 이 게인들을 용이하게 구할 수 있다. m_i는 속도응답이 최소의 오버슈트와 정착시간을 갖는 조건으로부터 간단한 시뮬레이션으로 구해진다. m_i의 설정에 관한 내용은 4장 수치 시뮬레이션에서 구체적으로 그 예를 보인다.

Fig. 2.6은 실제 제어계를 구성할 경우의 속도제어계의 블록도를 나타낸다. 가속도정보는 계측된 속도를 미분함으로써 구해지고, 미분연산은 다수의 고조파 성분을 동반하므로 시정수 T인 디지털 1차 LPF(Low Pass Filter)를 가속도 연산기에 추가한다. 이때 가속도연산기의 미분게인 K_1을 $K_1 = J/K_t$로 설정한다. 이렇게 함으로써 가속도정보를 피드포워드적으로 보상하면서도 기존의 전류제어기를 내부루프로 갖는 속도제어계와 등가적인 제어기의 설계가 가능하게 되어, 제안하는 방식과의 제어특성을 비교하는데 용이하게 하였다.

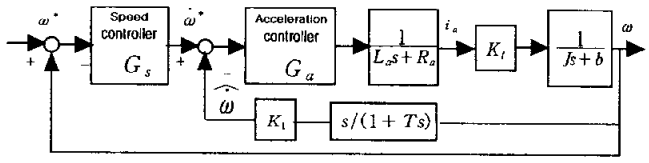


Fig. 2.6 Block diagram of speed control system including LPF and acceleration controller

2.3 4축 위치동기제어계의 설계

앞 절에서는 외란에 강인한 가속도제어기를 내부루프로 갖는 속도제

계가 설계되었다. 그러나, 각 축 모터의 동특성이 서로 다르고, 과도상태 및 외란이 인가되었을 경우는 각 축간에 속도응답 차이의 누적으로 인한 위치동기오차가 필연적으로 발생하게 된다. 따라서 과도상태를 포함한 전 운전영역에서 위치동기오차를 최소화할 수 있는 위치동기제어기를 설계한다.

우선 Fig. 2.7에 제안하는 4축 위치동기제어계의 구성을 보인다.

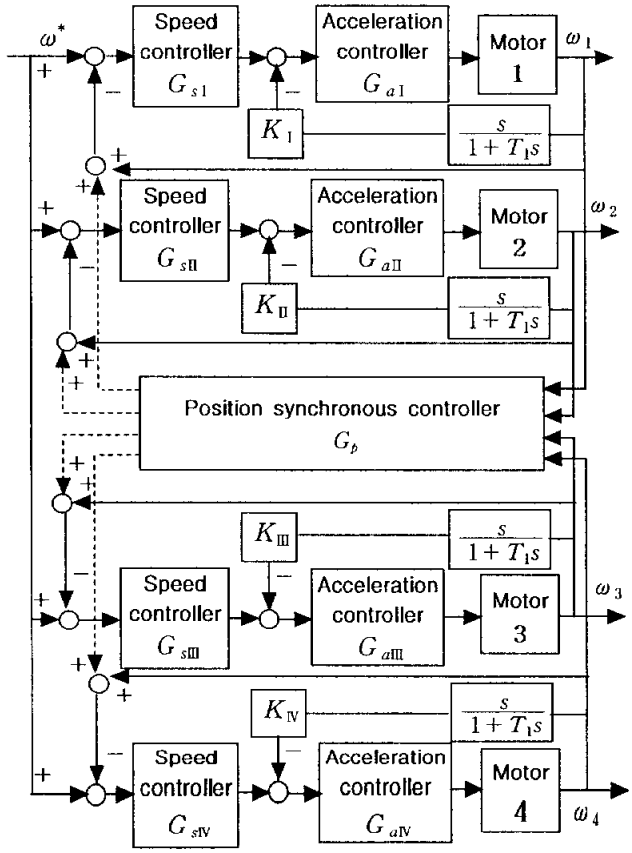


Fig. 2.7 block diagram of four-axes position synchronous control system

각 축은 우선 크게 제어기와 모터로 구성된다. 모터는 편의상 제 1축의 모터를 Motor 1, 제 2축의 모터를 Motor 2, 제 3축의 모터를 Motor

3, 제 4축의 모터를 Motor 4로 칭한다. 제어기는 모터의 기동과 정지를 포함한 전 운전영역에서 속도지령을 추종하기 위한 속도제어기, 부하 변동, 마찰토크 등의 외란을 신속히 보상하기 위한 가속도제어기로 구성되며, 속도제어기와 가속도제어기는 모두 PI 제어기가 이용된다. 가속도 정보는 엔코더로부터 계측된 속도정보의 미분연산에 의해 구해지며, 미분연산으로 인한 고조파 성분의 제거를 위해 1차 LPF를 통과시킨다. 그리고 속도제어기로의 지령은 속도지령과 각 축에 피드백되는 현재 속도정보와 위치동기제어계를 통과한 위치동기오차를 더한 값과의 차를 입력하게 된다. 위치동기제어계는 각 축간에 발생한 위치동기오차를 제어하여, 최대한 빠르게 위치동기를 취하도록 하기 위하여 구성된다. 위치동기제어계의 설계법으로는 기존의 2축 시스템에 적용되었던 협조제어방식을 수정하여, 4축 시스템에 적용이 가능하도록 최대 위치동기오차 비교법을 제안한다.

최대 위치동기오차 비교법은 각축의 속도정보를 샘플링 시간 동안 적분한 위치정보를 검출하여 나머지 세 축과의 위치정보를 축차 비교한 후, 오차의 절대치가 상대적으로 가장 큰 위치동기오차를 갖는 축과의 오차를 자신의 축에 우선적으로 보상하는 방법이다. 이 방법에 근거한 최대 오차의 크기는 식 (2.15)와 같이 나타낼 수 있다.

$$e_{pi} = \max\left[\int\{\omega_i(t) - \omega_j(t)\}dt\right] \qquad \textbf{(2.15)}$$

$$(\text{ for } i,j=1\sim4;\ i\neq j\ )$$

여기서, max는 비교값들 가운데 최대치를 취하는 함수를 나타낸다.

식 (2.15)를 좀더 구체적으로 설명하기 위해 각 축에 보상되는 최대오차를 산출해 본다. 이해를 돕기 위해 샘플링 시간 동안의 속도 적분치가 아닌, 속도지령 및 각 축의 현재 속도응답에 주목하여 최대의 위치동기오차를 개략적으로 나타내어 본다. 우선, 속도지령 및 각 축의 속도응답

을 Table 2.1과 같이 가정한다.

Table 2.1 Speed reference and response for example case

Speed reference	Speed response of each axis	
600 [rpm]	1 Axis(Motor 1)	530 [rpm]
	2 Axis(Motor 2)	600 [rpm]
	3 Axis(Motor 3)	560 [rpm]
	4 Axis(Motor 4)	610 [rpm]

제 1축의 경우, 1축을 중심으로 제 2, 3, 4축과 속도응답을 축차 비교해보면, 그 절대치가 가장 크게 발생하는 축은 제 4축임을 알 수 있다. 마찬가지로 나머지 제 2축, 제 3축, 제 4축의 경우도 다른 세 축과 축차 비교하여 그 절대치가 가장 큰 축을 검색해보면, 제 2, 3, 4축은 각각 제 1, 4, 1축이 해당됨을 알 수 있다. 따라서 각 축에 보상되는 최대 위치동기오차는 $e_{p1} \sim e_{p4}$와 같이 산출해 볼 수 있다.

$$e_{p1} = \int_0^T (530-610)\,dt \qquad \text{(2.16 a)}$$

$$e_{p2} = \int_0^T (600-530)\,dt \qquad \text{(2.16 b)}$$

$$e_{p3} = \int_0^T (560-610)\,dt \qquad \text{(2.16 c)}$$

$$e_{p4} = \int_0^T (610-530)\,dt \qquad \text{(2.16 d)}$$

여기서, T는 샘플링 주기를 나타내고 있다. 이때 그 위치동기 오차의 크기는 $-80T$, $70T$, $-50T$, $80T$의 순으로 됨을 알 수 있다. 결국, Fig. 2.7에 있어서 각 축에 보상되는 피드백 속도와 위치동기오차에 의해 각 축의 속도제어기에 지령하는 정보는 현재 지령속도에 가장 느린 응답을 보이는 제 1축이 가장 빠르게 가속하도록 하고, 그 다음 느린 응답인 제

3축이 가속하도록 하고, 두 번째로 빠른 응답을 보이는 제 2축은 감속하게 하고, 가장 빠른 응답을 보이는 제 4축은 다른 축과 빠른 동기를 취하기 위해 가장 크게 감속하도록 지령을 주게 된다. 이렇게 함으로써 궁극적으로 특정 축의 큰 속도희생 없이 네 축이 빠르게 위치동기를 실현하게 된다. 이와 관련된 구체적인 수치 시뮬레이션은 제 4장에서 M/S 방식과 비교해서 보인다.

다음으로, 편의상 두 축에 주목한 위치동기제어기 설계에 관해 검토한다. 먼저, 내부루프가 전류제어기인 경우와 가속도제어기인 각 경우에 대하여 외란 인가시의 위치동기오차의 크기를 상호 비교하여 고찰한다.

인가되는 외란을 $T_j(j=1,2)$로 하고 속도지령은 동일하며, 위치동기제어기를 갖지 않는 경우의 전류제어계 및 가속도제어계의 각 축의 속도는 각각 식 (2.17) 및 식 (2.18)과 같이 유도된다.

$$\omega_{jc}(s) = \frac{A_{1j}s + A_{2j}}{B_{1j}s^3 + B_{2j}s^2 + A_{1j}s + A_{2j}}\omega^*(s) - \frac{L_{aj}s^2 + K_{cpj}s}{B_{1j}s^3 + B_{2j}s^2 + A_{1j}s + A_{2j}}T_j(s) \tag{2.17}$$

$$\omega_{ja}(s) = \frac{C_{1j}Ts^2 + (C_{2j}T + C_{1j})s + C_{2j}}{B_{1j}Ts^4 + B_{1j}s^3 + (C_{1j}T + B_{3j})s^2 + (C_{1j} + C_{2j}T)s + C_{2j}}\omega^*(s) - \frac{L_{aj}Ts^3 + L_{aj}s^2}{B_{1j}Ts^4 + B_{1j}s^3 + (C_{1j}T + B_{3j})s^2 + (C_{1j} + C_{2j}T)s + C_{2j}}T_j(s) \tag{2.18}$$

여기서, 각 항은 다음과 같다.

$$A_{1j} = K_{tj}K_{cpj}K_{spj}\ ,\quad A_{2j} = K_{tj}K_{cpj}K_{sij}$$
$$B_{1j} = J_jL_{aj}\ ,\quad B_{2j} = J_jK_{cpj}\ ,\quad B_{3j} = J_jK_{apj}$$
$$C_{1j} = K_{tj}K_{apj}K_{spj}\ ,\quad C_{2j} = K_{tj}K_{apj}K_{sij}$$

위 식에서 $T_j=0$ 이고, 정상상태인 경우는 식 (2.17)과 식 (2.18)모두 최종치정리에 의해 $\omega_j=\omega^*$로 되며, 위치동기오차는 0으로 수렴된다. 그러

나, 인가되는 부하외란 T_j가 D_1, D_2의 크기를 가지는 스텝상의 외란으로 상정하여 고찰해 보면, 이때의 위치동기오차는 최종치정리를 적용함으로써 전류제어계는 식 (2.19)와 같이, 가속도제어계는 식 (2.20)과 같이 각각 유도된다.

$$\begin{aligned} e_{p\,c}(s) &= \lim_{s\to 0} s\frac{1}{s}\{\omega_1(s)-\omega_2(s)\} \\ &= -\frac{1}{K_{t1}K_{si1}}D_1+\frac{1}{K_{t2}K_{si2}}D_2 \end{aligned} \tag{2.19}$$

$$\begin{aligned} e_{p\,a}(s) &= \lim_{s\to 0} s\frac{1}{s}\{\omega_1(s)-\omega_2(s)\} \\ &= 0 \end{aligned} \tag{2.20}$$

식 (2.19)와 같이 전류제어계를 내부루프로 갖는 위치동기제어계에서는 스텝상의 부하외란이 인가되었을 경우, 순간적으로 큰 속도변동의 발생으로 그 시점부터 축간의 위치동기오차의 발생이 필연적이다. 또한, 그 크기는 D_1, D_2가 동상일 경우에는 서로 상쇄되는 효과가 있으나, 역상일 경우 가장 큰 값을 갖게 되며, 토크정수와 적분게인이 클수록 오차가 작게됨을 알 수 있다. 따라서, 이 경우는 위치동기오차를 0으로 수렴시키기 위한 위치동기제어기의 설계가 필수적이다. 하지만 식 (2.20)과 같은 가속도제어를 이용한 위치동기제어계는 전류제어계와 비교하여 외란인가시 현저하게 작은 속도변동이 발생하며, 위치동기오차도 0으로 수렴하게 된다. 따라서 이론적으로는 별도의 위치동기제어기의 설계없이도 정상상태에서 정밀한 위치동기를 실현할 수 있다. 하지만 본 연구에서는 외부 루프에 각 축간의 발생한 위치동기오차를 정상상태 뿐만 아니라 과도상태에서부터 더욱 속응성있게 제거하기 위해 최대 위치동기오차 비교법을 적용하고 있으므로 단위 피드백으로써 위치동기제어기를 비례게인 만을 갖는 K_{pp}=1의 가장 단순한 형태로 설계한다.

제 3 장 수치 시뮬레이션

3.1 제어계의 동적강성

서보계에 있어 외란토크에 대한 속도응답의 전달함수에서 전달함수의 절대값이 작을수록 그 제어계는 외란에 강인한 제어 특성을 갖게된다. 이때, 전달함수의 역수를 취하고 라플라스 연산자 $s=j\omega$를 이용하여 각속도 ω의 함수로 표시한 것을 동적강성(dynamic stiffness)라고 정의하면, 전류제어계와 가속도제어계를 내부루프로 갖는 단일 축 속도제어계의 블록도는 Fig. 3.1과 Fig. 3.2와 같고, 각 제어계의 동적 강성을 나타내는 전달함수는 각각 식 (3.1)과 식 (3.2)와 같이 구할 수 있다. (단, 마찰계수 $b=0$, 미분이득 $K_1=J/K_t$ 에 의해 $K_{cp}=K_{ap}$로 한다.)

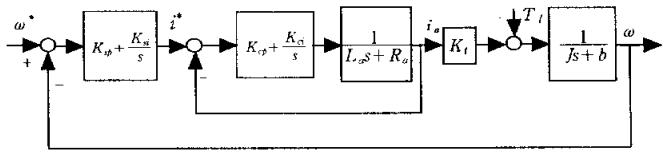


Fig. 3.1 Speed control system based on current control

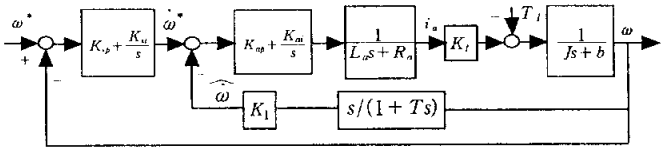


Fig. 3.2 Speed control system based on acceleration control

$$\left|\frac{T_l}{\omega_m}\right|_{cur}=\left|\frac{\alpha(j\omega)^3+\beta(j\omega)^2+\gamma(j\omega)+\delta}{L_a(j\omega)^2+K_{cp}(j\omega)}\right| \tag{3.1}$$

$$\left|\frac{T_l}{\omega_m}\right|_{accel} = \left|\frac{\alpha T(j\omega)^4 + \beta(j\omega)^3 + \{\gamma T + \beta\}(j\omega)^2 + \{\delta T + \gamma\}(j\omega) + \delta}{L_a T(j\omega)^3 + L_a(j\omega)^2}\right| \quad (3.2)$$

여기서, $\alpha = JL_a$, $\beta = JK_{ap}$, $\gamma = K_{sp}K_{ap}K_t$, $\delta = K_{ap}K_{si}K_t$ 이다.

식 (3.1)과 식 (3.2)에서 알 수 있듯이, 외란의 각 주파수 ω가 낮은 부분(느린 주파수의 외란)에서는 동적강성은 전류제어계와 가속도제어계 모두 δ에 의해 결정되며, 직류($\omega = 0$)에서 무한대의 강성을 갖게 된다. 즉, 일정한 크기의 외란 토크에 대해서는 정상상태에서 두 제어계 모두 속도오차가 없는 것을 의미하지만, 빠른 주파수의 외란이 시스템에 인가되는 경우는 동적강성이 J에 크게 영향을 받게 된다. 이에, 가속도제어계를 취함으로써 전류제어계에 비하여 얼마만큼의 동적강성이 증대되는 것인지, 식 (3.1)과 식 (3.2)의 주파수특성을 알아보기 위해, 보드선도의 게인선도를 구하여 그 차이를 확인해 보았으며, 그 특성은 Fig. 3.3과 같다.

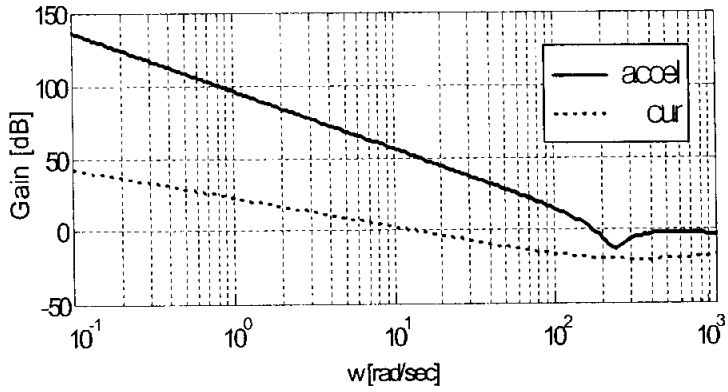


Fig. 3.3 Gain characteristic of bode diagram to confirm dynamic stiffness

$\omega = 0.1 \sim 1000[rad/s]$의 넓은 주파수 범위에서 가속도제어계가 전류제어계에 비하여 게인 값이 크게 나타나고 있다. 따라서 가속도제어계가 그 상대적인 크기만큼의 동적강성이 더 크다고 볼 수 있다.

3.2 제어기의 설정

본 장에서는 제안된 제어칙과 모터의 실제 파라미터를 이용하여 수치 시뮬레이션을 통해 제어기를 설계한다. 먼저, 단일축의 속도제어계 시뮬레이션에서는 가속도제어기, 속도제어기를 설계한 후 전류제어계와 비교하여 그 속도응답을 확인한다. 그리고, 4축 위치동기제어계의 시뮬레이션에서는 본 연구에서 제안하는 가속도제어를 내부루프로 가지며 최대 위치동기오차 비교법을 통한 위치동기제어법과 기존의 전류제어계를 갖는 방식 및 M/S 방식의 위치동기제어법과의 비교 시뮬레이션을 통해 제안 방식의 유효성을 확인한다. 시뮬레이션은 MATLAB에서 수행되었으며, Table 3.1은 시뮬레이션에 사용된 모터의 파라미터를 나타낸다.

Table 3.1 Motor parameters and rated values

Term		Axes		Unit
		Motor 1, 2	Motor 3, 4	
Rated Power	P_R	300	200	[W]
Rated Torque	T_R	0.95452	0.637	[Nm]
Rated Speed	N_R	3000	3000	[rpm]
Rated Current	I_R	4.8	3.3	[A]
Rated Voltage	E_R	75	75	[V]
Back Electro-Motive Force Constant	K_e	0.2333	0.2262	[V/rpm]
Torque Constant	K_t	0.22246	0.2156	[Nm/A]
Inertia of Rotor	J	2.45×10^{-4}	1.76×10^{-4}	[Nmsec^2]
Resistance of Armature	R_a	1.02	1.53	[Ω]
Inductance of Armature	L_a	1.07×10^{-3}	1.75×10^{-3}	[H]

가속도제어기 설계시 교차각주파수 ω_c에 의해 비례게인과 적분게인이 정해짐을 2장 2.2절에 기술하였다. 따라서, 가속도제어계의 속응성은 ω_c를 크게 하면 전류제어기의 게인에 비례해서 증가함을 알 수 있다. 하지만 ω_c의 값은 앞서 밝힌 것처럼 디지털계에서의 실현일 경우, 샘플링주

기의 영향과 하드웨어 성능의 제한으로 인해 무한히 크게 하는 것은 불가능하다. 현재, 이 값의 적절한 크기를 결정하는 방법은 명확히 해명되어 있지 않은 상태이다. 단지, 앞서 밝힌 바와 같이 연속계에서의 실현을 고려할 경우, ω_c는 PWM(Pulse Width Modulation)쵸퍼 캐리어주파수의 $2\pi/3$이하로 제한되는 것으로 알려져 있으므로, 본 논문에서는 연속계에서의 ω_c의 10배 이하로, 즉 PWM 쵸퍼의 캐리어주파수의 $2\pi/30$이하로 제한하기로 한다. 실험장치의 PWM 쵸퍼의 캐리어주파수는 15.625[KHz]이므로, ω_c는 3,272[*rad/s*]로 설정하였다. 이 경우 가속도제어계의 개루프 전달함수 및 폐루프 전달함수의 주파수 특성은 각각 Fig. 3.4 및 Fig. 3.5와 같다.

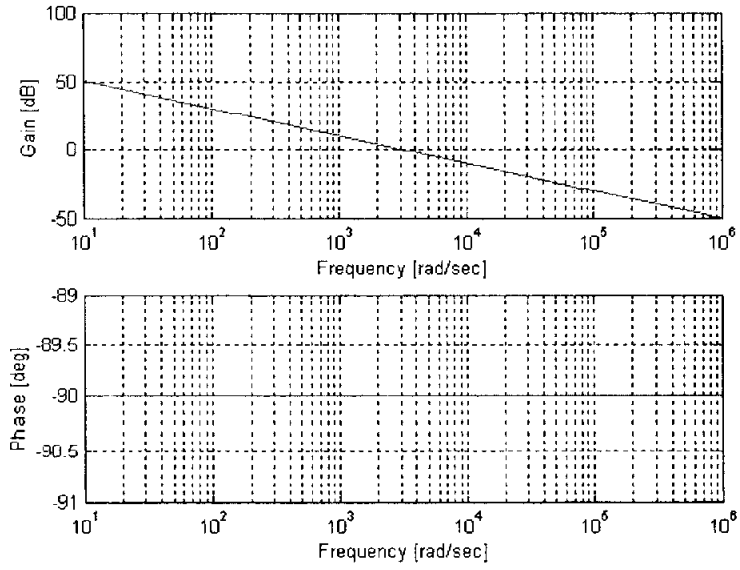


Fig. 3.4 Open loop frequency characteristic of acceleration control system

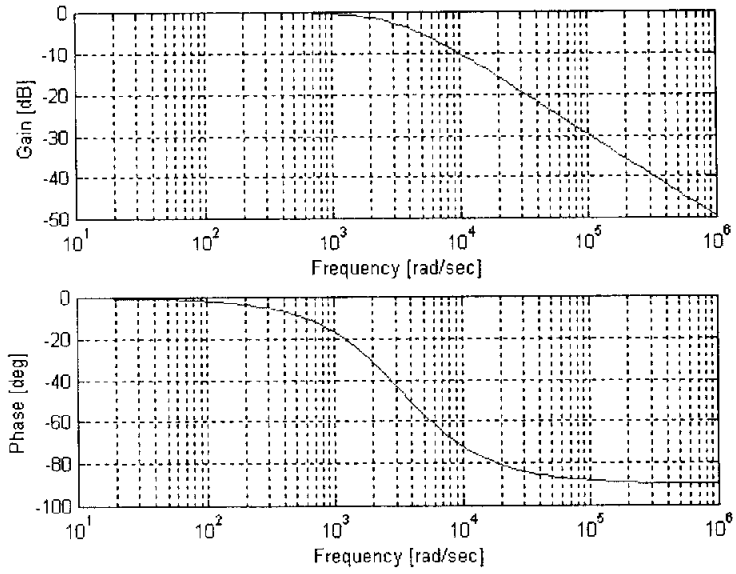


Fig. 3.5 Closed loop frequency characteristic of acceleration control system

Fig. 3.4에서 위상각이 항상 -90[°]로 극히 안정한 계임을 알 수 있으며, Fig. 3.5를 통해 단순 1차 요소의 특성임을 알 수 있고, 교차각주파수에서 -45[°]의 위상각을 가지며, 정상상태에서 0[dB]의 게인을 갖는 안정한 특성을 보임을 알 수 있다.

3.3 가속도제어계의 안정성

앞의 제2장에서 가속도제어기는 전기계의 시정수를 적분시간으로 설정함으로써 교차각 주파수와 모터의 파라미터에 근거하여 비례게인을 용이하게 설정할 수 있었다. 하지만, 실제의 서보시스템에 적용시에는 구동시의 내부 파라미터 변동 등으로 인해, 모터 모델의 파라미터를 모터 설계 사양치로 상정하여 설계된 제어기는 그 성능이 양호하게 계속 유지된다고는 보기 어렵다. 특히, 모터의 이너셔와 토크상수는 주변의 환경적 조건 및 모터의 회전자의 위치에 따라 공칭치가 엄밀하게 달라지게 된다.

게다가 모터의 전기자 저항은 장시간 운전으로 발생하는 내부 열에 의해 그 값이 증가하기 쉬운 요소이다. 따라서, 이와 같은 파라미터 변동시에도 앞서 설계한 가속도제어계가 성립하는지를 검토해 본다. 해석상 편의를 위해, 가속도연산기의 미분이득은 $K_1 = J/K_t + \Delta\alpha$ (단, $\Delta\alpha = \pm 0.5 \cdot J/K_t$)로 하고, 저항값이 초기 설정치 보다 100[%] 증가한 상태를 가정한다. 이 경우에는 식 (2.6)에서 T_{ai}의 간략화가 이루어지지 않게 되고, 그때의 가속도제어계의 개루프 및 폐루프 전달함수는 각각 식 (3.3)과 식 (3.4)로 된다.

$$G_a^o = \frac{K_{ap}(J + K_t\Delta\alpha)s + (J + K_t\Delta\alpha)K_{ai}}{(JL + 2RJT)s^2 + (2RJ + JL)s} \tag{3.3}$$

$$G_a^c = \frac{K_{ap}(J + K_t\Delta\alpha)s + (J + K_t\Delta\alpha)K_{ai}}{(JL + 2RJT)s^2 + (2RJ + JL + K_{ap}J + K_{ap}K_t\Delta\alpha)s + (J + K_t\Delta\alpha)K_{ai}} \tag{3.4}$$

그리고 이 때의 폐루프 전달함수의 주파수응답은 Fig. 3.6에 나타내었다. 그림에서 알 수 있듯이 그 주파수 특성은 가속도제어기의 미분이득 및 모터 파라미터가 변동한 경우에도 교차각주파수 $\omega_c = K_{ap}K_t / L_a J$ 부근에서 -45[°]의 위상각을 가지며, 정상상태에서 0[dB]의 게인 값을 갖는 안정한 특성을 보인다.

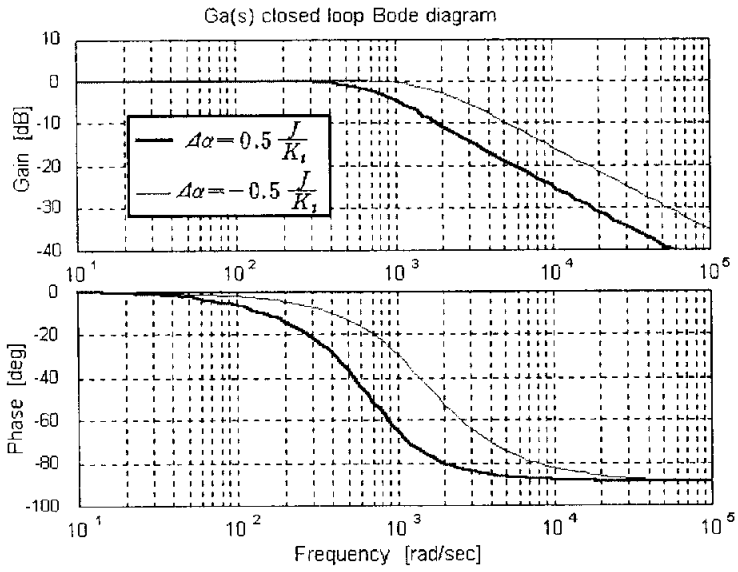


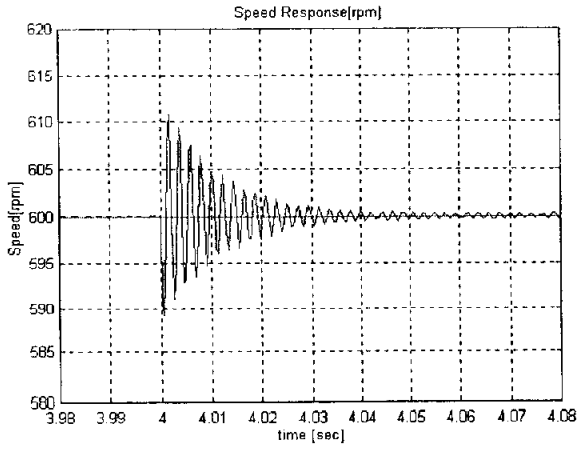
Fig. 3.6 Frequency characteristic of acceleration control system under various conditions

속도제어기의 비례게인 및 적분게인은 ω_c에 대한 적절한 ω_{sc}와 ω_{pi}를 설정함으로써 용이하게 구할 수 있다. 하지만 적절한 m_1, m_2의 설정을 위한 뚜렷한 기준이 없으므로 이하에서는 수치 시뮬레이션을 이용하여 각 값들에 대한 속도응답을 비교 검토함으로써 m_1, m_2를 설정하도록 한다.

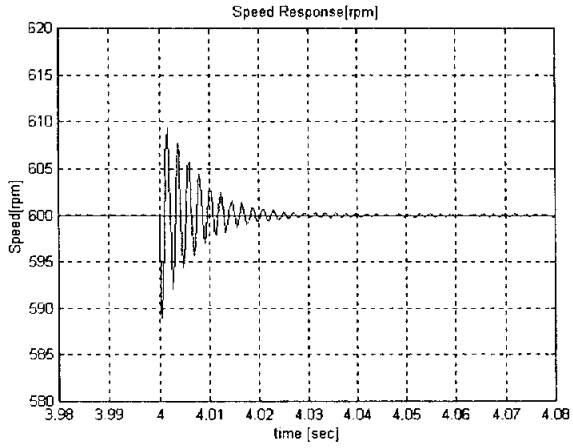
Fig. 3.7은 $\omega_{sc} = \omega_c / m_1$의 관계에서 $m_1 = 2, 5, 10$ 의 세가지 경우에 대해 $\omega_{pi} = \omega_{sc} / m_2$에서의 $m_2 = 2, 5, 10$의 각각의 조합으로, 총 9가지의 경우에 대해 시뮬레이션을 수행하였다. 그러나, $m_1 = 2$, $m_2 = 2$인 경우는 시스템이 발산하여 나머지 8가지 경우에 대한 속도응답만을 보인다.

속도지령은 램프적으로 기동으로부터 3[sec] 이내에 600[rpm]에 도달한 후 정속도를 유지하도록 인가하였으며, 시각 4[sec]의 시점에 제 4축 모터에 정격토크의 100[%]에 해당하는 스텝상의 토크외란을 인가하였다.

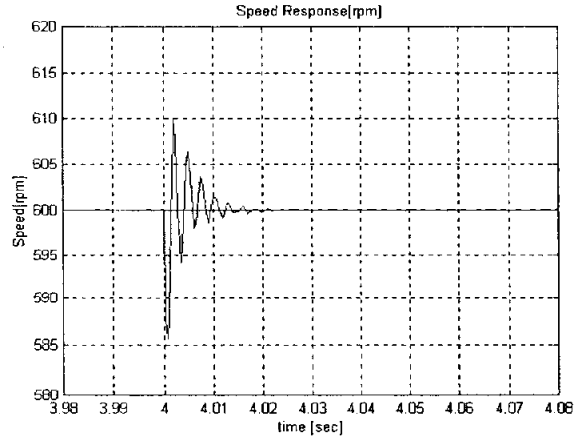
Fig. 3.7에서 (a)~(b)는 $m_1=2$에 대하여, $m_2=5$, 10일 경우의 속도응답을 각각 나타내고 있으며, (c)~(e)는 $m_1=5$에 대하여, $m_2=2$, 5, 10일 경우의 속도응답을 각각 나타내고 있다. 마찬가지로, (f)~(h)는 $m_1=10$에 대하여, $m_2=2$, 5, 10일 경우의 속도응답을 각각 나타내고 있다. 결과에서 알 수 있는 바와 같이 가속도연산으로 인하여 속도응답에 다소 리플 성분이 나타나지만, (a)~(h)중에서 리플 성분이 가장 작고, 과도한 오버슈트와 언더슈트가 발생하지 않으며, 정착시간이 가장 빠른 경우는 $m_1=m_2=5$인 경우임을 알 수 있다. 따라서, 본 연구에서의 시뮬레이션 및 실험에서는 속도 제어기의 설정에 있어서 $\omega_{sc}=\omega_c/m_1$와 $\omega_{pi}=\omega_{sc}/m_2$에서의 m_1, m_2를 $m_1=m_2=5$로 정하였다.



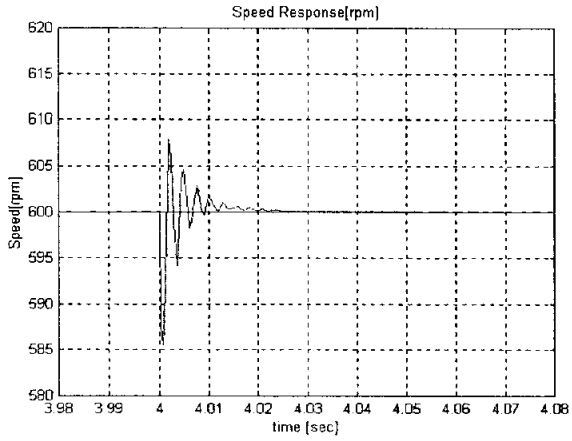
(a) $m_1=2$, $m_2=5$



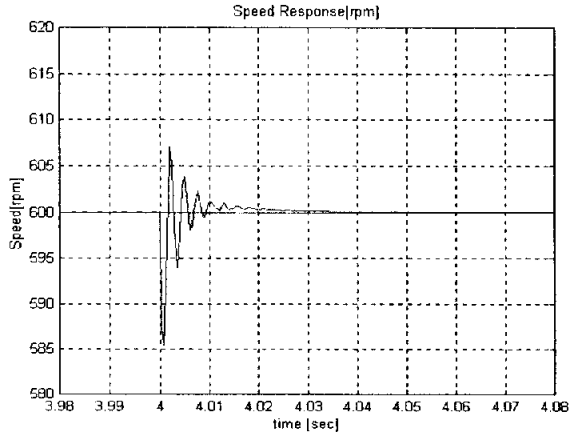
(b) $m_1 = 2,\ m_2 = 10$



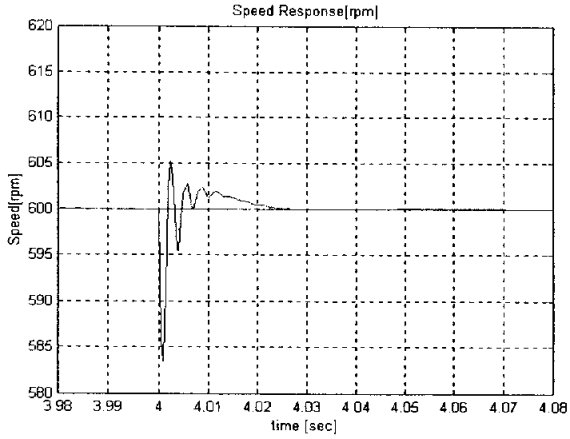
(c) $m_1 = 5,\ m_2 = 2$



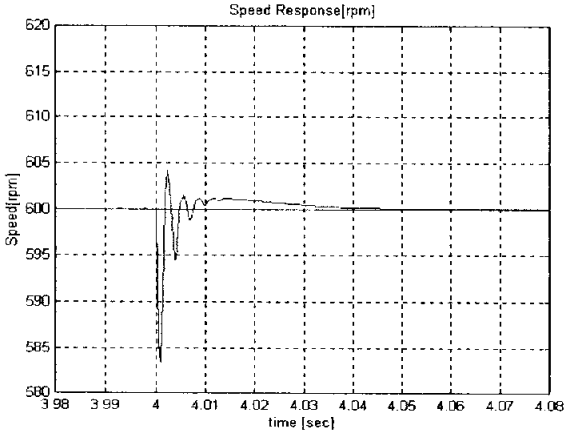
(d) $m_1 = 5,\ m_2 = 5$



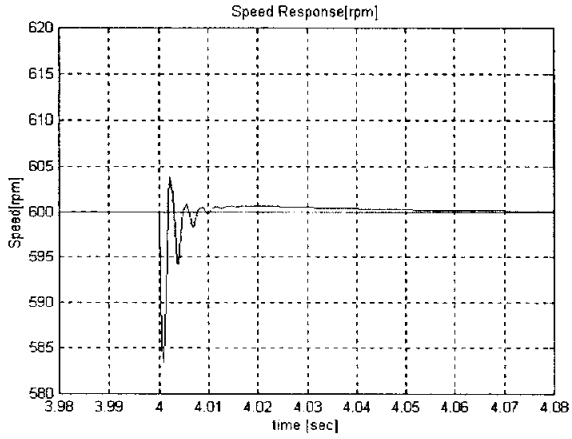
(e) $m_1 = 5,\ m_2 = 10$



(f) $m_1 = 10,\ m_2 = 2$



(g) $m_1 = 10,\ m_2 = 5$



(h) $m_1 = 10,\ m_2 = 10$

Fig. 3.7 Speed responses on relation of between ω_{sc} and ω_{pi} under disturbance torque

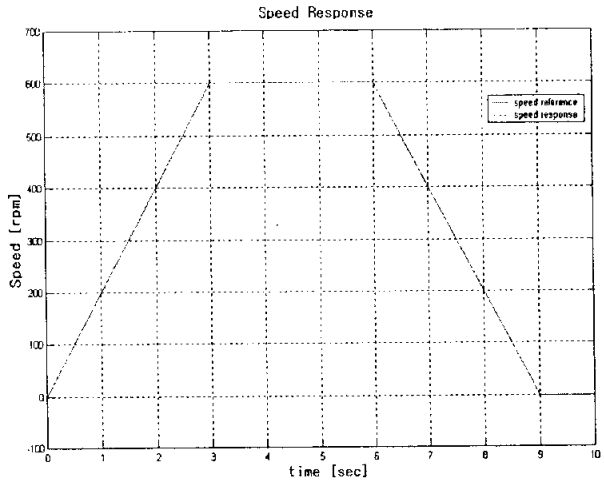
3.4 단일축의 제어 응답특성

앞장에서 설계한 각 제어기와 대상 모터의 파라미터를 이용하여 수치 시뮬레이션을 행한다. 시뮬레이션은 MATLAB 프로그램에서 수행하였으며, 앞서 설정한 제어기의 설계법에 따른 제어기의 게인은 Table 3.2와 같다.

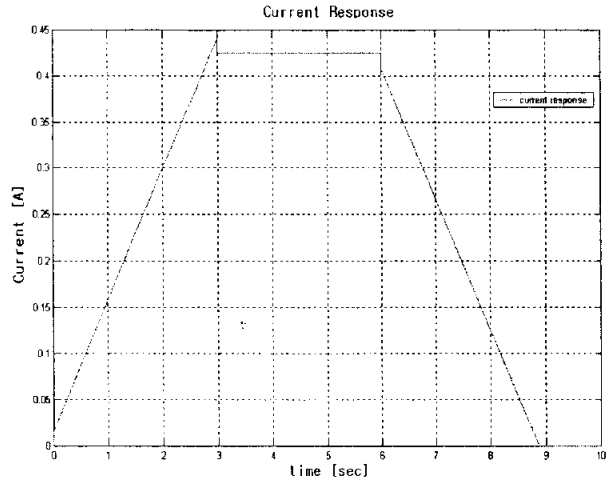
Table 3.2 Value of controller parameter

Parameter		Value	
		1, 2 axis	3, 4 axis
Current Controller & Acceleration Controller	K_{cp} , K_{ap}	3.5	5.721
	T_{ci} , T_{ai}	1.04×10^{-3}	1.143×10^{-3}
Speed Controller	K_{sp}	0.72	0.529
	K_{si}	93.6	68.7
Position Synchronous Controller	K_{pp}	1	1

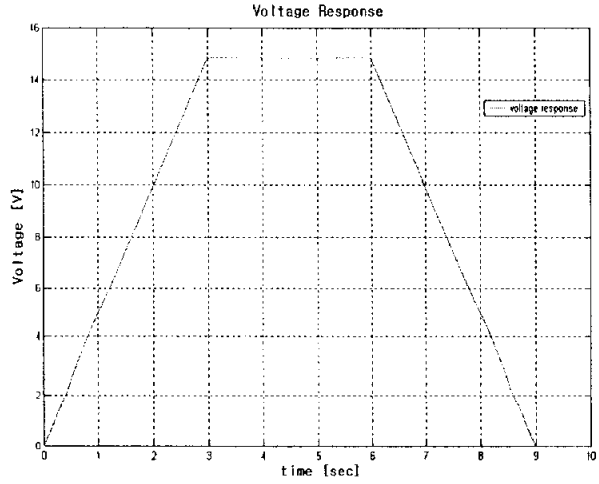
Fig. 3.8은 단일 모터에 대하여 가속도제어를 갖는 속도제어계의 수치 시뮬레이션의 한 결과로, 속도지령은 3[sec]동안 600[rpm]에 도달하도록 인가하였을 때, 속도제어계의 응답을 보여 주고 있다. 여기서, (a)는 속도 응답, (b)는 전류응답, (c)는 전압지령을 나타내고 있다. 제안된 알고리즘은 과도상태를 포함한 전 운전영역에서 속도지령을 정확하게 추종하고 있음을 알 수 있다.



(a) speed response



(b) current response



(c) voltage reference

Fig. 3.8 Simulation result of single-axis motor based on acceleration control

다음으로, 외란인가 하의 가속도제어계 및 전류제어계의 속도응답의 비교 결과를 Fig. 3.9에 보인다. 속도지령은 3[sec]동안 600[rpm]에 도달하도록 인가하였고, 4[sec] 시점에 제 4축 모터(Motor 4)에 정격토크의 100[%]에 해당하는 스텝상의 부하토크 외란을 인가하였다.

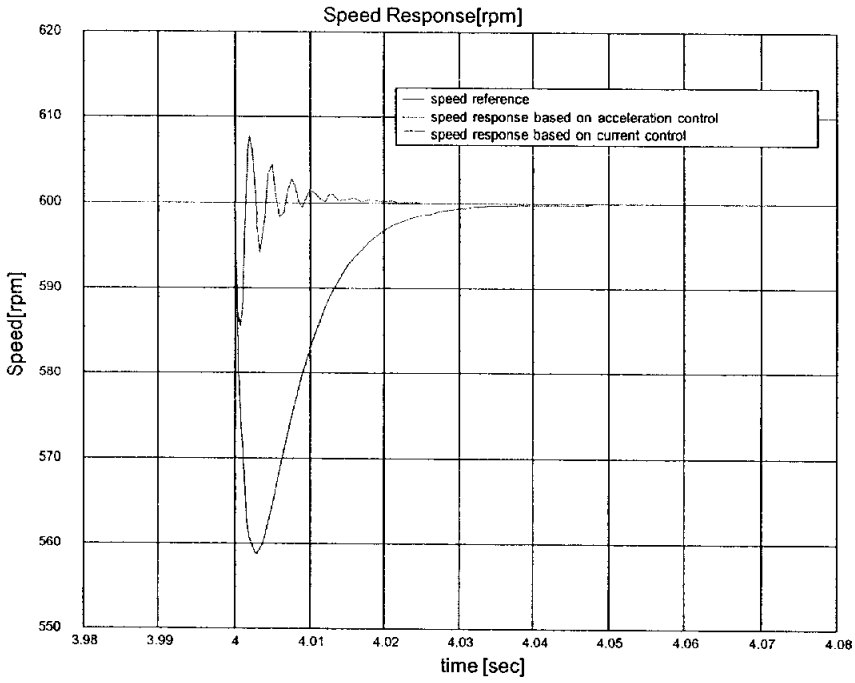


Fig. 3.9 Speed response of single-axis motor based on acceleration and current control under step load disturbance

Fig. 3.9에서 알 수 있는 바와 같이 제안된 방식이 전류제어계에 비하여 작은 언더슈트 및 빠른 정착시간을 가지며, 외란에 로버스트하게 동작하고 있으므로 가속도제어방식이 더욱 유효하다고 볼 수 있다.

Fig. 3.10은 Fig. 3.9과 같은 시뮬레이션 조건으로, 외란이 스텝상이 아

닌 정현파상의 외란을 인가했을 경우의 속도응답이다. 속도지령은 Fig. 3.9의 조건과 동일하고, 4[sec] 시점에 제 4축 모터(Motor 4)에 정격토크의 100[%]에 해당하는 크기와 10[Hz]주파수를 갖는 정현파 형태의 부하토크 외란을 인가하였다.

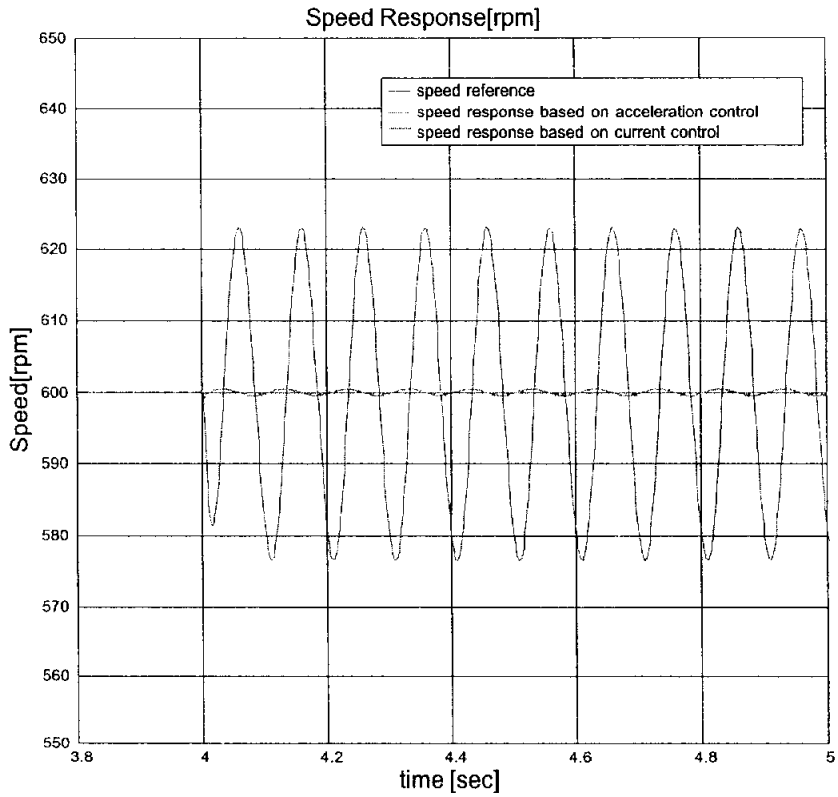


Fig. 3.10 Speed response of single-axis motor based on acceleration and current control under sinusoidal load disturbance

3.5 4축 위치동기제어계의 시뮬레이션 및 고찰

4축 위치동기제어계의 시뮬레이션에서는 특히 전류제어기와 가속도제어기로 구성된 위치동기제어계에서 각 제어기의 외란 제거 성능에 따른 위치동기오차의 크기에 주목하였다. 속도지령은 램프적으로 2[sec]동안 600[rpm]에 도달하도록 지령하였고, 5[sec] 시점에서 10[sec]까지 제 4축 모터(Motor 4)에 정격토크의 50[%] 및 100[%]에 상당하는 스텝상의 토크 외란을, 그 외의 시간에는 모터의 브레이크로 인한 마찰토크에 상당하는 크기의 외란을 각각 인가하였다.

Fig. 3.11과 Fig. 3.13은 전류제어기를 이용한 경우의 위치동기 제어 시뮬레이션 결과를 나타내고 있다. Fig. 3.11은 정격토크의 50[%] 크기의 스텝상의 외란이 인가되었을 경우의 결과를, Fig. 3.13은 정격토크의 100[%] 크기의 외란이 인가될 경우의 속도 및 전류응답, 부하외란 및 위치동기오차의 크기를 각각 나타내고 있다. 이 결과를 통해 외란이 인가되더라도 속도응답은 속도지령에 잘 추종하고 있음을 알 수 있으나, 외란이 인가되는 순간 미소한 속도 차로 인해 가장 큰 위치동기오차가 발생되고 있음을 확인할 수 있다.

Fig. 3.12와 Fig. 3.14는 앞의 전류제어계와 동일한 속도지령 및 외란인가 조건 하에서 제안한 가속도제어기를 이용했을 경우의 위치동기제어 시뮬레이션 결과를 나타낸다. 결과에서 알 수 있듯이 모터의 정격토크 범위까지 부하외란이 인가될 경우에도, 전류제어계에 비하여 제안된 방식이 우수한 외란제거 성능으로 빠르게 위치동기를 취하며, 위치동기오차의 크기 역시 최소화되고 있음을 알 수 있다.

본 시뮬레이션은 전류제어기와 가속도제어기를 내부루프로 갖는 위치동기제어계의 외란 제거 성능을 단순 비교하기 위한 것으로, 두 방식 모두 최대 위치동기오차 비교법에 의한 동일한 위치동기제어기를 갖는 것으로 가정하였다.

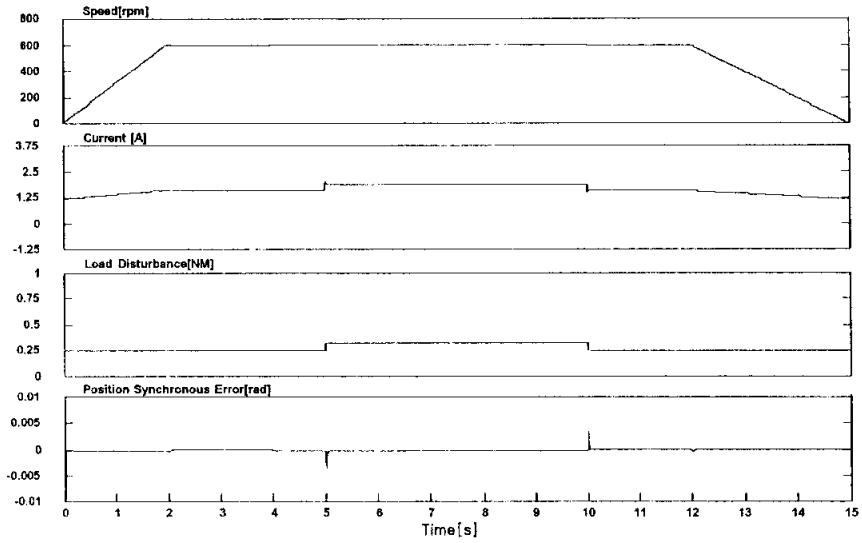


Fig. 3.11 Simulation result under 50[%] load torque of motor 4 based on current control system

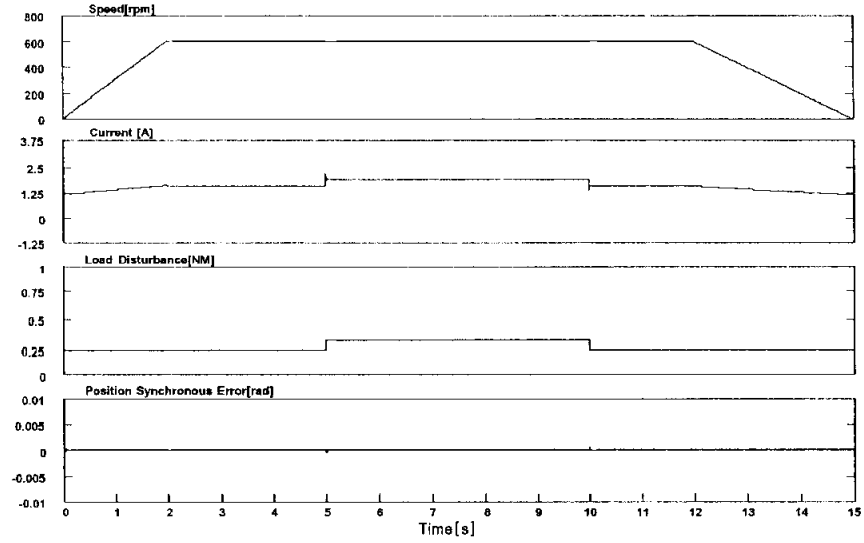


Fig. 3.12 Simulation result under 50[%] load torque of motor 4 based on acceleration control system

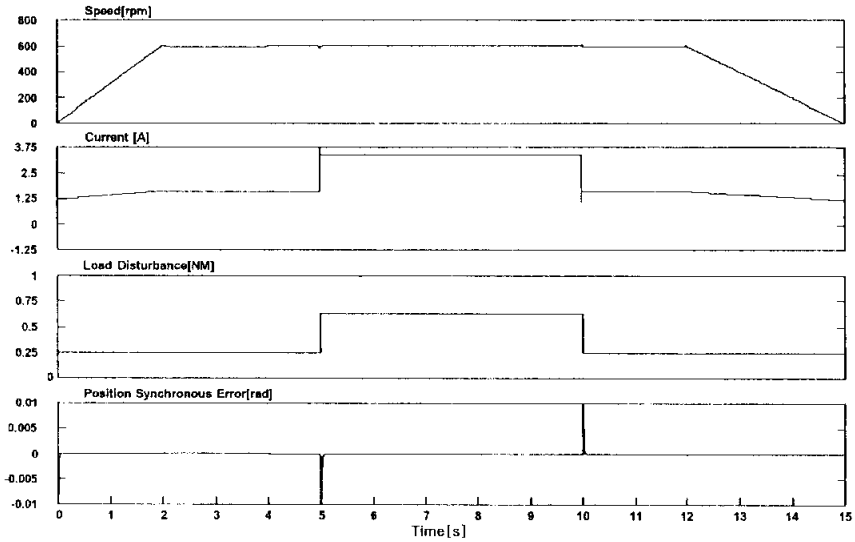


Fig. 3.13 Simulation result under 100[%] load torque of motor 4 based on current control system

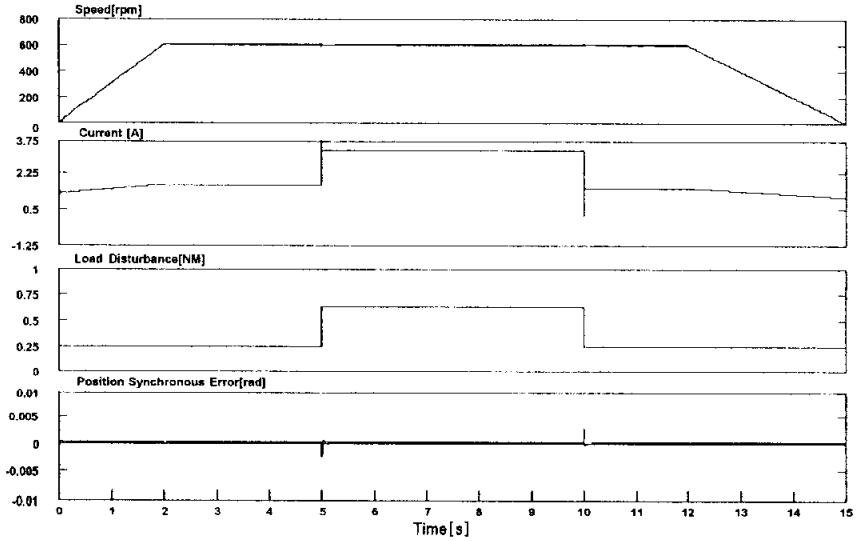
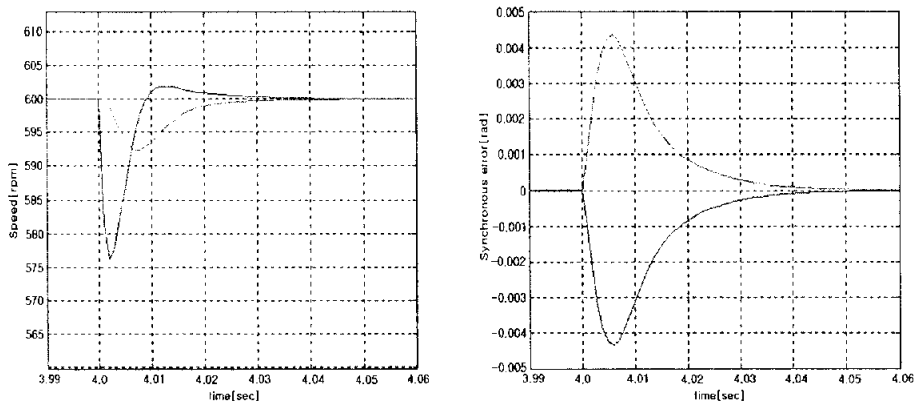


Fig. 3.14 Simulation result under 100[%] load torque of motor 4 based on acceleration control system

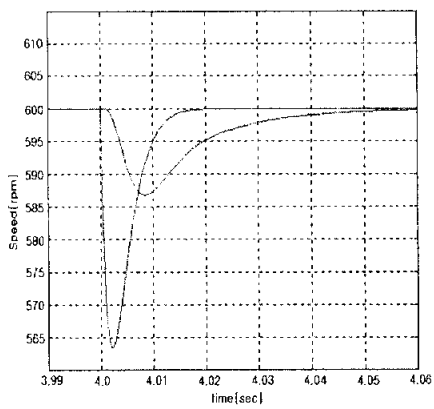
다음은 4축 시스템에 대해, 최대 위치동기오차 비교법과 M/S 방식과의 비교 시뮬레이션 결과를 보인다. 속도지령은 램프적으로 4[sec]동안 600[rpm]에 도달하도록 지령하였고, 4[sec] 시점에서 제 4축 모터(Motor 4)에 정격토크의 50[%] 크기의 스텝상의 외란 토크를 인가하였다.

Fig. 3.15는 최대 위치동기오차 비교법에 의한 속도응답 및 위치동기오차 파형을, Fig. 3.16은 M/S 방식에 의한 결과를 나타내고 있다. 파형에서 알 수 있는 바와 같이 최대 위치동기오차 비교법이 M/S 방식에 비해 외란이 인가된 축의 속도 언더슈트도 더 작으며, 정착시간도 더 빠름을 알 수 있다. 즉, 최대 위치동기제어 방식이 외란이 인가된 축에 주목하여 일방적인 속도 희생을 감수해야 하는 M/S 방식에 비하여, 더 빠른 속도 지령치 추종응답을 보인다. 결국, 속도정보의 누적에 의한 위치동기오차 크기도 더 작게 나타나고 있어 제안하는 최대 위치동기제어 방식이 더 유효함을 확인할 수 있다.

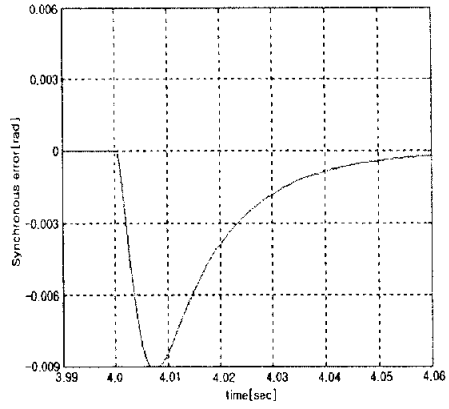


(a) Speed response (b) Position synchronous error

Fig. 3.15 Simulation result based on maximum position synchronous error comparison



(a) Speed response



(b) Position synchronous error

Fig. 3.16 Simulation result based on master-slave method

제 4 장 실기실험 및 고찰

4.1 하드웨어의 구성 및 특징

Fig. 4.1은 4축 시스템의 위치동기제어 실험을 위해 구성된 하드웨어 블록도이며, Photo. 4.1은 실제 실기실험을 위해 자체 구축한 하드웨어 실험장치를 나타낸 것이다.

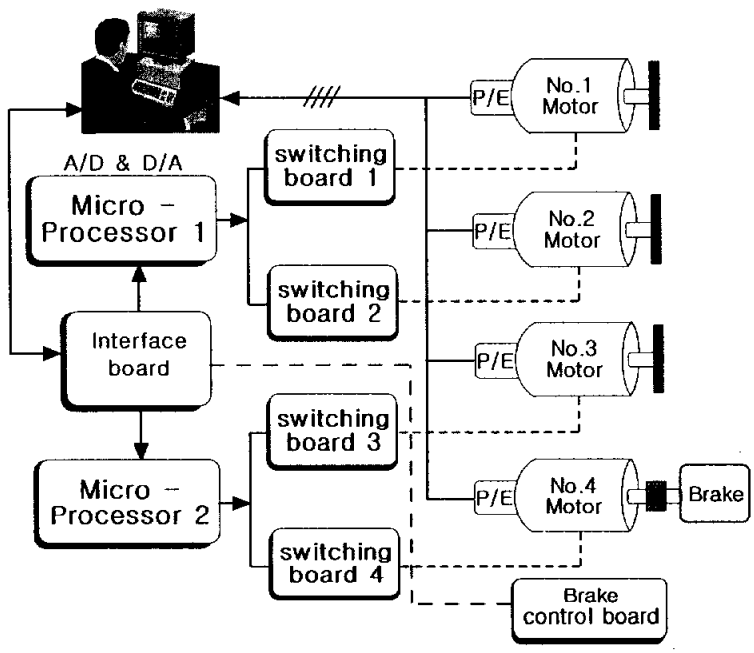


Fig 4.1 Schematic diagram of experimental system

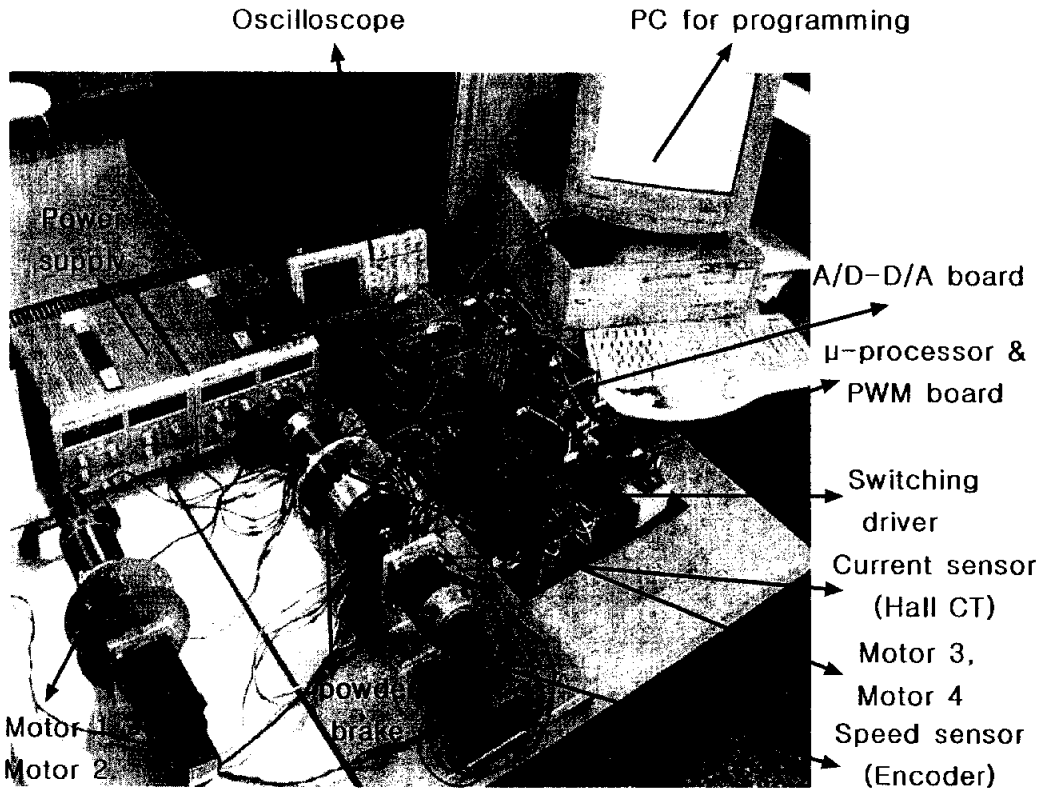


Photo. 4.1 Hardware devices for experiments

전체적인 하드웨어 특징은 다음과 같다.

① PWM 구현을 위한 마이크로프로세서는 인텔사의 80c196kc를 사용하였다. 80c196kc에는 총 3개의 PWM 포트가 있으며, 본 실험에서는 PC에서 연산된 전압지령을 PWM0, PWM1 포트를 이용하여 256의 DUTY비로 출력한다. 내부 클럭주파수로 16MHz를 사용하였으며 이때, PWM 주기는 64μs(≒ 15.6[㎑])이다.

② 스위칭보드는 파워 트랜지스터를 이용해 자작한 선형 서보증폭기이며, 파워 트랜지스터로는 전류구동형인 MJ11016을 사용하였다.

③ 모터의 전류검출은 20[A]/5[V]의 홀센서(Hall CT)를 사용하였다. 그러나, 실제 실험에서는 전류검출의 정도를 향상시키기 위해 홀센서에 3번 감아 4번 통과하는 방법으로 5[A]/5[V]의 스케일로 맞추어 사용하였다. 본 연구에서 제안하는 방식의 비교실험을 위하여 모터의 전류정보는 전류제어형 실험에서는 피드백 전류정보로서, 가속도제어에서는 모터 및 제어장치의 과전류 보호용으로 이용된다.

④ 속도검출은 모터에 기본적으로 장착되어 있는 1000 [pulse/rev.]의 엔코더를 사용하였다. 엔코더 펄스의 계수를 위한 엔코더 카운트보드는 4-input 카운트보드를 사용하였고, 분해능 증가를 위해 4체배 기능을 사용하여 결국 4000[pulse/rev.]의 정보를 이용하여 속도를 연산하였다.

⑤ 가속도정보는 엔코더의 정보로부터 연산한 속도정보를 시간에 대한 미분연산으로 구하였다. 하지만 미분연산은 그 특성상 다수의 고조파 성분을 동반한다. 따라서 고조파 제거를 위해 간단한 1차 LPF를 프로그램상에서 구현하여 가속도 연산에 추가하였다.

⑥ A/D, D/A보드를 이용하여 홀센서에서 검출된 아날로그 전류검출

값을 PC로 A/D 변환시키고, 연산된 전압지령을 D/A 변환시켜 스위칭 보드로 인가하였다. 이 보드는 12bit 분해능의 16ch. A/D포트와 12bit 분해능의 2ch. D/A포트로 구성되었으며, 8bit의 DI(Digital input), 8bit의 DO(Digital output) 기능이 있다. 신호의 입출력은 외부의 인터페이스보드를 이용하였다.

⑦ 제어프로그램은 펜티엄급 PC에서 Turbo-C를 이용하여 구현하였다. 또한 Turbo-C의 그래픽 기능을 이용하여 각 정보의 실시간 모니터링이 가능하도록 프로그램 하였다.

⑧ 외란 인가장치는 파우더 브레이크(powder brake)를 제 4축의 모터축에 커플링을 통해 연결하였으며, 브레이크의 기본 사양은 Table 4.1과 같다. 실험에서는 스텝상의 부하외란이 Fig. 4.2와 같이 구성된, 자작한 제어보드를 통하여 원하는 시점에 원하는 크기의 토크를 발생할 수 있도록 하였다.

고안한 제어보드와 브레이크에 인가한 소스전압 대비 발생토크를 알기 위해 기초실험을 통해 얻은 연산토크 및 외란토크 추정 알고리즘[부록 A-2]을 이용한 관측 토크, 그리고 브레이크 제작사 카다로그 상의 메뉴얼 토크의 3가지 데이터 값을 비교해서 Fig. 4.3에 나타내었다.

그림에서 알 수 있듯이 실험 및 관측기를 통해 추정한 토크 값이 비교적 유사하게 나와서, 실험에서는 외란 관측기 이론을 바탕으로 하여 그 값에 근거한 부하외란을 소정의 소스전압을 통하여 정량적으로 인가하였다.

Table 4.1 Powder Brake parameters

Model	정격토크	정격전압	정격전류	사용 회전수
NKPB-0.3	2.94[N·M]	DC 24[V]	0.68[A]	1000[rpm] 이내

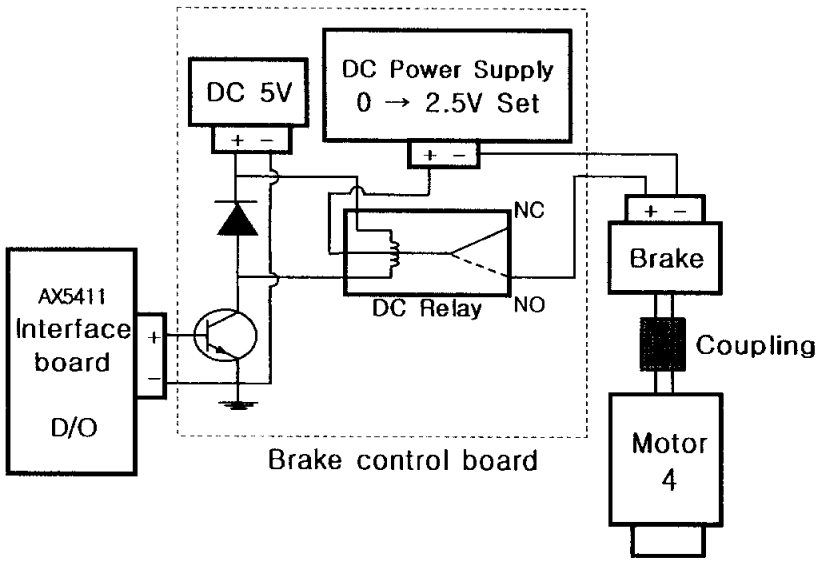


Fig. 4.2 Control board of powder brake

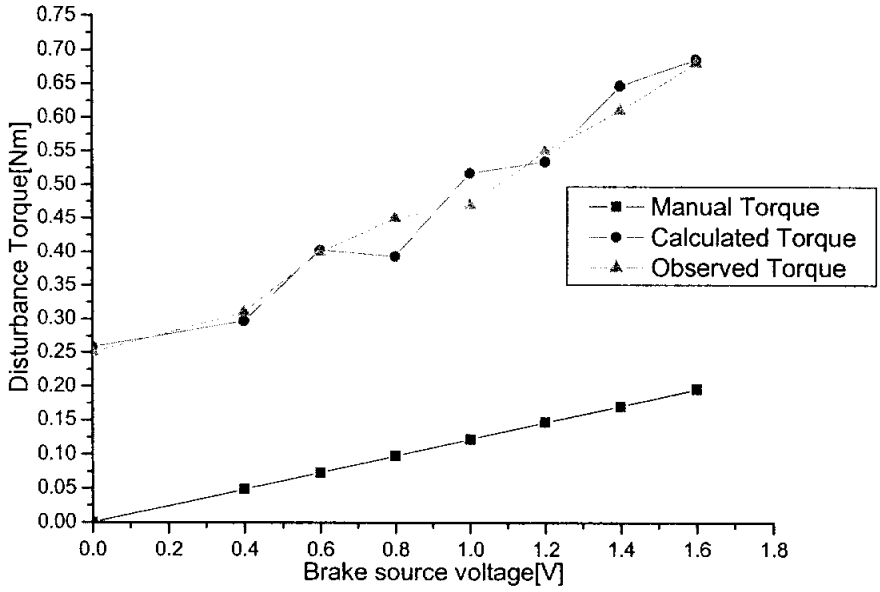


Fig. 4.3 Disturbance torque characteristic of powder brake

4.2 소프트웨어의 구성

Fig. 4.4는 C 프로그램에서 구현된 기존의 전류제어계를 갖는 제어프로그램의 순서도를 나타내고, Fig. 4.5는 본 연구에서 제안하는 가속도제어계를 갖는 제어프로그램의 순서도를 나타낸 것이다.

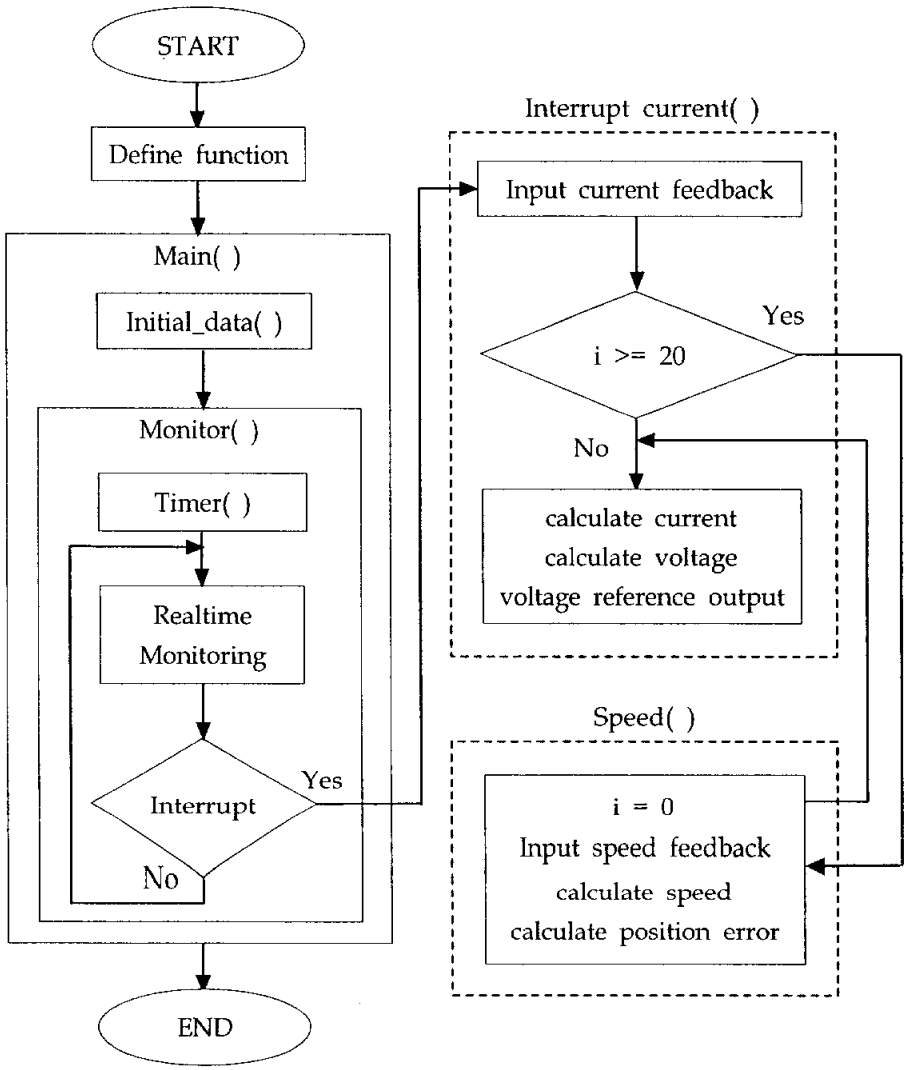


Fig. 4.4 Program flow chart for experimental program based on current control

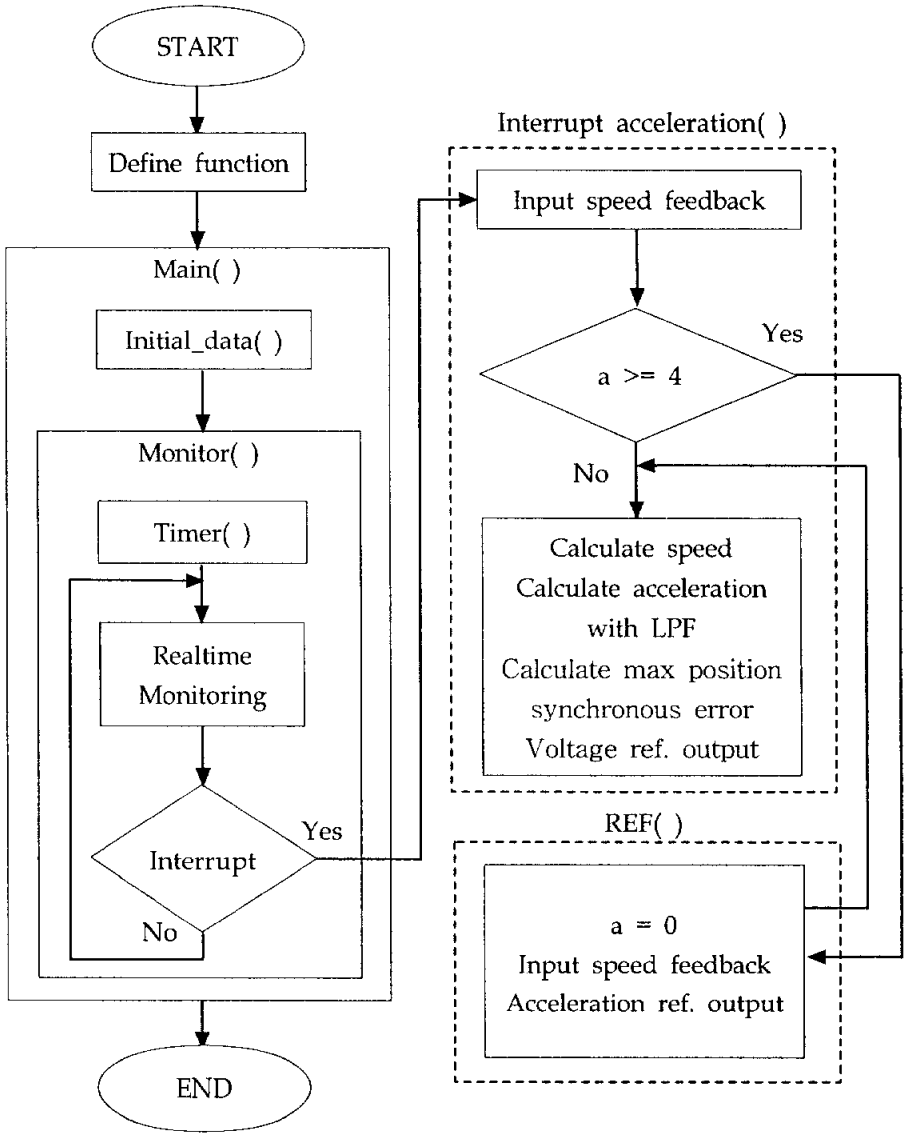


Fig. 4.5 Program flow chart for experimental program based on acceleration control

가속도제어계의 제어프로그램은 크게 3부분으로서 메인 루틴, 속도·가속도 연산 및 최대 위치동기오차 산출 루틴, 속도 보상 지령 및 가속도 지령 루틴으로 구성되어 있으며 각각의 기능은 다음과 같다.

① 메인루틴에서는 변수 정의, 초기치 설정, 인터럽트시간 등을 정의하고 인터럽트 시간 외의 무한루프 중에는 연산된 변수들을 C 프로그램의 그래픽 기능을 이용하여 파형으로 실시간 모니터를 통해 출력한다. 프로그램이 실행되면 출력하고자 하는 변수에 따라 메뉴에서 원하는 프로그램을 지정한 후 속도지령을 입력하면 시간에 따라 변수값을 출력하게 된다.

② 설정된 인터럽트 시간이 되면 가속도 연산 루틴으로 와서 모터의 엔코더에서 출력되는 펄스 수를 카운트하여, 그 정보로 현재의 각 축의 속도를 연산하고 각 축간의 속도차의 적분으로 위치동기오차를 연산한다. 그리고 검출된 속도정보는 미분연산과 LPF 처리를 통해 가속도정보로 연산되고 가속도제어기를 이용해 전압지령을 생성한다. 전압지령은 적절한 스케일링을 거쳐 인터페이스 보드의 D/A포트와 DO포트를 이용하여 마이크로프로세서에 인가된다. 전압지령의 출력은 하나의 마이크로프로세서보드에서 2축의 제어가 가능하도록 프로그램을 작성하였으며 이 알고리즘은 다음과 같다. 즉, PC에서 연산된 전압지령 중 한 축의 전압지령은 아날로그 값으로 80c196kc의 A/D변환을 이용하여 입력되고, 다른 한 축의 전압지령은 인터페이스보드를 이용한 8bit의 디지털값으로 80c196kc의 port 1을 통해 입력된다. 이렇게 입력된 두 전압지령은 80c196kc 보드에 장착된 ROM에 미리 입력된 프로그램에 의해 실시간 PWM 정보를 출력한다. 80c196kc의 ROM 내부에 작성한 PWM 발생 프로그램은 부록[A-3]에 수록하였다. 가속도제어 루틴 샘플링주기는 500 [μs]로 설정하였다.

③ 속도 피드백과 가속도 지령 루틴에서는 이전의 루틴에서 연산한 속도와 최대 위치동기오차를 각 축에 피드백 보상하고, 이 보상값과 속도 검출값을 이용해 속도오차를 연산하고 속도제어기를 거쳐 가속도지령을 생성한다. 이 루틴은 모터의 총합적인 가속도 변동을 가속도 연산루틴에서 빠르게 보상하고 있으므로, 가속도연산 루틴 4번에 1회의 속도 연산이 이루어져 지령하게 된다. 따라서, 샘플링주기는 2[㎳]로 설정하였다.

4.3 실험결과 및 고찰

실험에서도 수치 시뮬레이션과 마찬가지로 본 논문에서 제안한 방식과 기존의 전류제어기를 내부 루프로 갖는 위치동기제어법과의 비교 실험을 수행하였다. 외란 인가장치는 Photo. 4.1에서와 같이 제 4축(Motor 4)에만 설치되어 가변 토크 브레이크 및 자체 제작한 제어회로를 통해 원하는 시점에 소정의 소스전압을 브레이크에 인가함으로써 필요한 크기의 외란이 인가되도록 하였다.

Fig. 4.6에서 Fig. 4.9까지는 동일한 조건 하에서 전류제어기와 가속도 제어기를 내부루프로 갖는 경우의 성능을 대비시킨 실험결과의 예이다. 두 경우 모두, 기동 후 3[sec]에 600[rpm]의 램프상의 속도지령이 실현되도록 설정하였다. 또한, 5[sec] 시점에서 10[sec]까지 Motor 4에 정격토크의 50[%] 및 정격토크에 해당하는 스텝상의 토크 외란을 각각 인가했을 경우의 속도와 전류응답, 관측기를 통한 부하외란 추정 및 위치동기오차를 각각 나타내고 있다.

Fig. 4.6과 Fig. 4.7은 정격토크의 50[%] 크기의 외란 인가시, Fig. 4.8과 Fig. 4.9는 100[%] 크기의 외란이 인가된 경우의 위치동기제어 결과를 각각 나타내고 있다. 그림에서 속도는 속도지령과 네 축의 속도응답을, 전류치는 제 4축만의 응답을 각각 나타내고 있다. 특히, 위치동기오차는 제 4축과 나머지 3개의 축들 가운데 최대 위치동기오차를 갖는 축과의 데이터를 나타내고 있다. Fig. 4.6에서 Fig. 4.9에서의 전류응답은 속도의 미분 연산을 통한 가속도 정보의 고조파 성분으로 인해 약간의 리플 성분

이 나타나고 있음을 알 수 있다. 그리고 제안한 방법의 실험결과인 Fig. 4.7과 Fig. 4.9의 응답이 전류제어계인 Fig. 4.6과 Fig. 4.8의 결과에 비하여 위치동기오차가 약 1/3배 작게 나타나고 있음을 알 수 있다. 통상 위치동기오차의 크기는 사용 엔코더의 속도분해능에 제약을 받게 되는데, 한 샘플링 시간에 최대 1펄스의 에러를 허용하는 것으로 하면, 이로 인해 발생되는 위치오차의 크기는 p가 엔코더의 1회전당 펄스 수일 경우 $2\pi/p \fallingdotseq 1.57\times10^{-3}$[*rad*]이다. 만약 각 축이 동시에 이와 같은 오차를 갖는 것으로 가정하면 최대 위치동기오차는 $4\pi/p$로서 p=4000일 경우 3.14×10^{-3}[*rad*]이다. 제안된 방식의 최대 위치동기오차가 2.97×10^{-3}[*rad*]이므로, 이는 엔코더의 분해능 근방에서 제어되고 있음을 알 수 있다.

본 논문에서 제안한 설계방법은 각 축이 서로 다른 용량의 모터로 구성된 경우뿐만 아니라, BLDC 서보모터를 이용한 다축 구동 시스템에서도 그 일반성을 잃지 않고 적용이 가능하다.

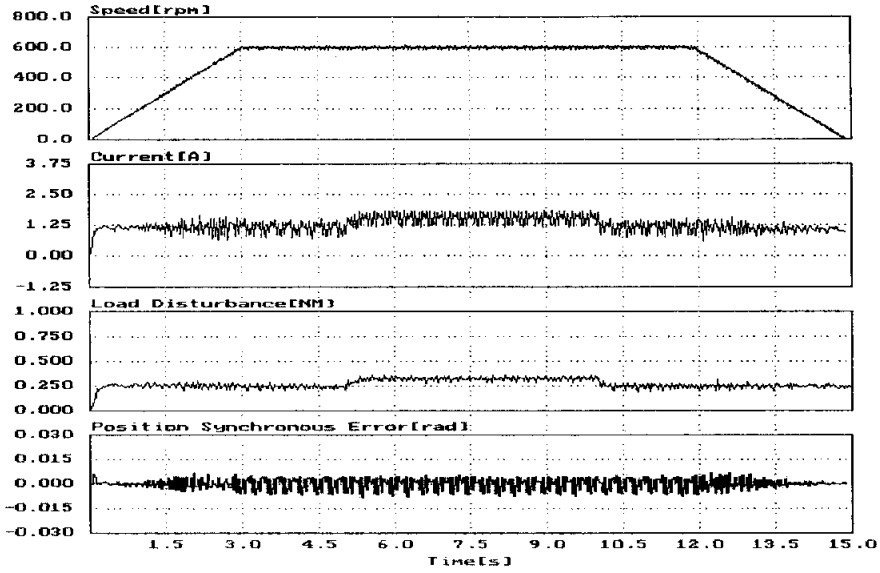


Fig. 4.6 Experimental result under 50[%] step disturbance of motor 4 based on current control system

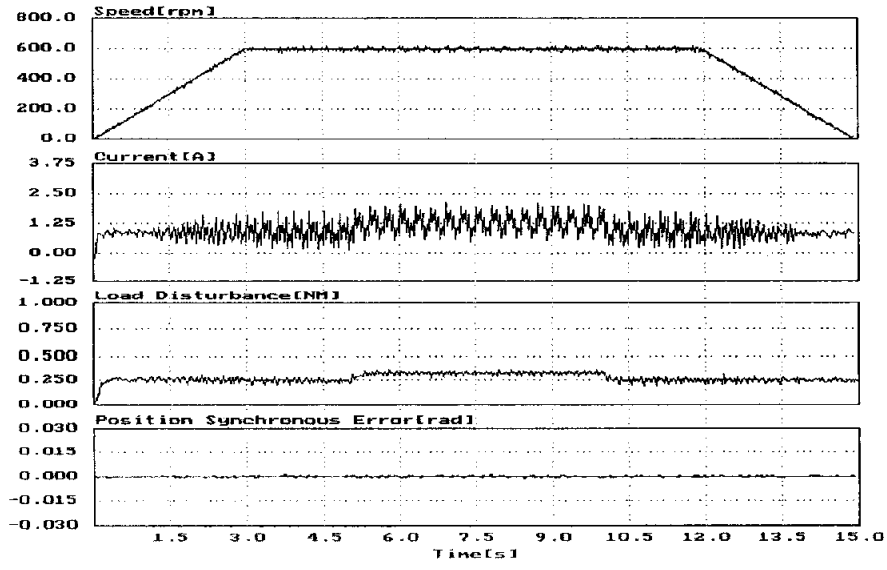


Fig. 4.7 Experimental result under 50[%] step disturbance of motor 4 based on acceleration control system

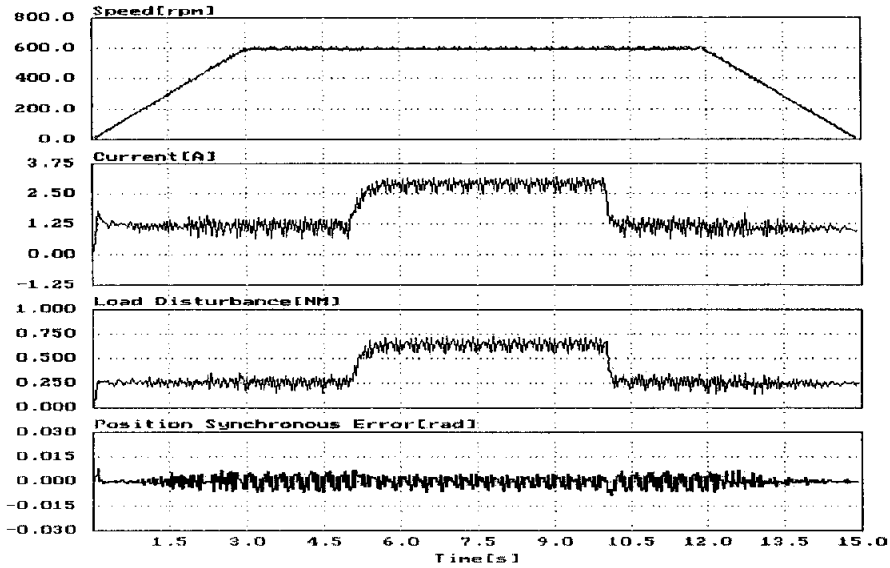


Fig. 4.8 Experimental result under 100[%] step disturbance of motor 4 based on current control system

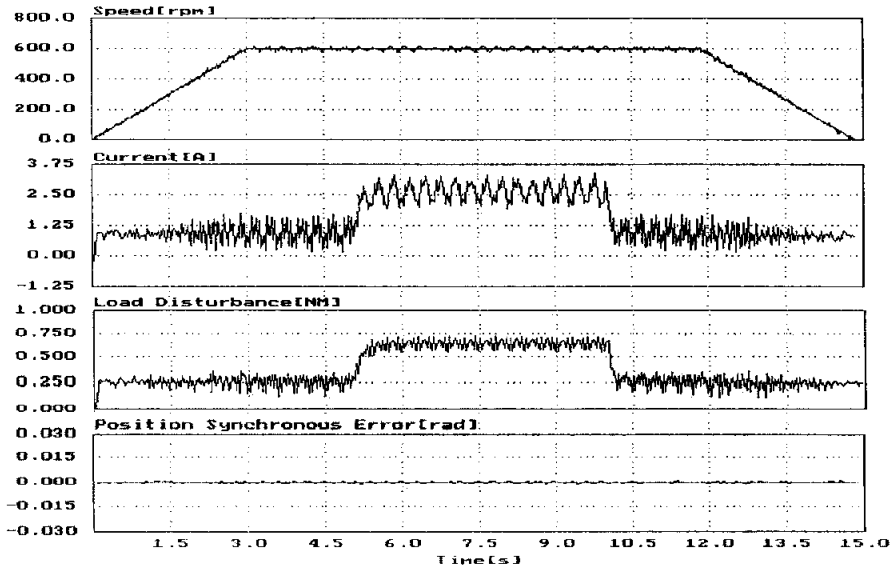


Fig. 4.9 Experimental result under 100[%] step disturbance of motor 4 based on acceleration control system

제 5 장 결 론

본 연구에서는 부하외란이 인가되는 실제적인 상황을 고려한 4축 시스템의 정밀 위치동기제어 방안이 제안되었다. 제안된 방안은 모터에 외란이 인가될 경우, 우선적으로 내부루프에 설계된 가속도제어기를 통해 이를 강력히 억제한다. 다음으로 각 축이 갖는 동특성의 차로 인해 발생되는 위치동기오차는 최대오차비교 알고리즘을 이용한 위치동기제어기를 통해 최소화되도록 하였다. 이로써 특정 축의 속도를 일방적으로 희생시키지 않고 고정도의 위치동기를 실현할 수 있음을 수치 시뮬레이션을 통해 입증하였다. 특히, 시뮬레이션에서는 기존의 전류제어기를 갖는 경우와 대비시켜 가속도제어기의 외란에 대한 강인성을 확인하였다.

정량적인 외란 인가가 가능한 실기실험 시스템을 제작하고 실제 외란 인가 실험을 통해 제안된 방식의 유효성을 확인하였다. 실험에서도 전류제어기를 갖는 경우와 대비시켜 가속도제어기를 갖는 위치동기제어계의 제어성능을 비교, 고찰하였다.

제안된 방식은 제어대상을 모터의 전기계의 회로방정식 및 기계계의 동역학을 이용해 모델링함으로써 체계적인 설계가 가능하도록 하였다. 또한, 현장에서 보편적으로 사용되고 있는 PID 제어칙에 근거한 설계법을 제시함으로써 현장에서의 적용이 용이하도록 배려하였다. 뿐만 아니라 각 제어기의 게인 설정도 시행착오 과정을 최소화 할 수 있는 설계법을 제시하였다. 부가적으로 가속도제어계에 있어서는 파라미터 변동으로 야기될 수 있는 계의 안정성에 대한 검토를 행하였다. 그리고 수치 시뮬레이션 및 실기실험을 통하여 얻은 주요 결과를 최종적으로 요약하면 다음과 같다.

(1) 가속도 제어기는 기존의 전류제어기에 비해 보다 우수한 외란 제거 성능을 보였다.

(2) 최대 오차 비교방식을 통한 위치동기제어기의 적용으로 전 운전 영역에서 빠른 속도추종과 정밀한 위치동기의 실현이 가능하였다.

(3) 정격토크에 상응하는 외란 인가시에도 최대 위치동기오차를 사용 엔코더의 분해능 수준인 2.97×10^{-3}[*rad*]이내로 제어할 수 있었다.

부 록

A-1 증분형 엔코더를 이용한 속도 분해능 계산

속도분해능 계산법

(1) $\omega = \frac{d\theta}{dt}$ 에 의해서 $[\text{rps}] = \frac{\omega}{2\pi}$

따라서, $[\text{rpm}] = \frac{\omega}{2\pi} \times 60 = \frac{d\theta}{T_c} \cdot \frac{60}{2\pi} = \frac{60X}{2\pi T_c}$

(2) 1회전당 P(PPR : Pulse Per Revolution)의 Encoder가 샘플링시간 T_c 시간 동안 m개의 Pulse 발생

• 각변위 X

1 회전 : $2\pi\ [rad] = \frac{m}{P}$ 회전 $: X\ [rad]$

$$\therefore X = \frac{2\pi}{P} m[rad]$$

$$\therefore N_f = \frac{60X}{2\pi T_c} = \frac{60}{2\pi T_c} \cdot \frac{2\pi}{P} m = \frac{60 \cdot m}{P T_c}$$

A-2 외란토크 추정 알고리즘

1. 등가외란 observer를 이용한 외란토크 추정

$$J\dot{\omega} + T_l = T_m \tag{1}$$

여기서, J : motor shaft inertia, T_l : load torque, T_m : motor torque를 나타낸다.

$$T_l = T_{inrt} + T_{ext} + T_{frc} \tag{2}$$

여기서, T_{inrt} : inertial torque, T_{ext} : external torque, T_{frc} : frictional torque를 나타낸다.

$$T_m = K_t i_a \tag{3}$$

여기서, K_t : torque constant, i_a : motor current

식(1) 및 식(2)에 의해

$$J\dot{\omega} = K_t i_a - (T_{inrt} + T_{ext} + T_{frc}) \tag{4}$$

식(4)에서 파라미터는 J 와 K_t 2개 존재하므로, 식(5)와 같이 나타낼 수 있다.

$$\begin{aligned} J &= J_n + \Delta J \\ K_t &= K_{tn} + \Delta K_t \end{aligned} \tag{5}$$

여기서, 첨자 n 은 nominal 값을 표시한다.

즉, $\Delta J\dot{\omega}$ 는 변화된 관성토크를, $\Delta K_t i_a$는 토크리플로 간주할 수 있다.
식(5)에 의해 외란토크는 식(6)과 같이 표현할 수 있다.

$$\begin{aligned} T_{dis} &= T_l + \Delta J\dot{\omega} - \Delta K_t i_a \\ &= T_{inrt} + T_{ext} + T_{frc} + (J - J_n)\dot{\omega} + (K_{tn} - K_t) i_a \end{aligned} \tag{6}$$

따라서, 식(3)에 의해 식(6)은 식(7)과 같이 된다.

$$T_{dis} = K_{tn}\, i_a - J_n \dot{\omega} \quad \Rightarrow \quad \hat{T}_l \tag{7}$$

2. Reduced-order asymptotic observer를 이용한 외란토크 추정

외란 토크의 크기가 Unknown이라면,

$$z = T_l + g\omega \tag{8}$$

여기서 z는 매개변수, g는 정수의 observer 게인을 나타낸다.

만약, 부하외란이 시간함수에 대해 느리게 변한다고 가정하면, $\dot{T}_l = 0$이라 둘 수 있으므로, 식(9)와 같이 나타낼 수 있다.

$$\dot{z} = g\dot{\omega} = g\left(\frac{1}{J} K_t i_a - \frac{1}{J} T_l\right) \tag{9}$$

식(8)을 식(9)에 대입하여 정리하면, 식(10)이 된다.

$$\dot{z} = \frac{1}{J}\left(-gz + g^2\omega + gK_t\, i\right) \tag{10}$$

z에 대해 observer를 설계하면, 식(11)로 표현되고,

$$\dot{\hat{z}} = \frac{1}{J}\left(-g\hat{z} + g^2\omega + gK_t\, i\right) \tag{11}$$

$\bar{z} = \hat{z} - z$ 라 두면, 식(10)과 식(11)의 관계에 의해 식(12)와 같이 된다.

$$\dot{\bar{z}} = -\frac{g}{J}\bar{z} \tag{12}$$

$g > 0$ 인 정수의 적절한 값의 선택으로 $\bar{z}$는 지수적으로 0의 값으로 수렴하게 된다.

결국, $\hat{z} = z$의 관계가 되므로, 추정토크는 식(13)과 같이 구할 수 있다.

$$T_l = \hat{z} - g\omega \tag{13}$$

A-3 PWM 발생 프로그램

```
$INCLUDE (80196ROM.INC)

PWM_1           EQU     16H
PWM_2           EQU     17H
IOC3            EQU     0CH
DT_1            EQU     20H
DT_2            EQU     22H
DUTY_1          EQU     24H
DUTY_2          EQU     26H
REG_1           EQU     28H
REG_2           EQU     2AH
REG_3           EQU     2CH

CSEG            AT      2080H
                LD      SP,#0C0H
                LDB     IOC0,#10100000B
                LDB     IOC1,#00000001B
                LDB     IOC2,#00000100B
                LDB     WSR,#01H
                LDB     IOC3,#00001100B
                LDB     WSR,#00H

ADSTART:
                LDB     ADCOM,#00011000B
ADBUSY:
                JBS     ADRESLO,3,ADBUSY
LOAD_DUTY_1:
                LDB     DT_1,ADRESHI
                CMPB    DT_1,#00H
                JNH     L_LIMIT_1
                CMPB    DT_1,#0FFH
                JH      H_LIMIT_1
LOAD_DUTY_2:
                LDB     REG_1,P1
                LDB     REG_2,P2
                ANDB    REG_1,#11100111B
                ANDB    REG_2,#00011000B
                ADDB    DT_2,REG_1,REG_2
                CMPB    DT_2,#00H
                JNH     L_LIMIT_2
                CMPB    DT_2,#0FFH
                JH      H_LIMIT_2
                SJMP    LOAD
L_LIMIT_1:
                LDB     DT_1,#00H
                SJMP    LOAD_DUTY_2
H_LIMIT_1:
                LDB     DT_1,#0FFH
                SJMP    LOAD_DUTY_2
L_LIMIT_2:
                LDB     DT_2,#00H
                SJMP    SETT
H_LIMIT_2:
                LDB     DT_2,#0FFH
SETT:
                LDB     DUTY_1,DT_1
                LDB     DUTY_2,DT_2
PWM ACT:
                LDB     WSR,#01H
                LDB     PWM_1,DUTY_1
                LDB     PWM_2,DUTY_2
                LDB     WSR,#00H
                SJMP    ADSTART

                END
```

A-4 4축 위치동기제어 시뮬레이션 프로그램

```
% Condition Initializer
=============================================

Wc=3272;                % 가속도제어계의 교차각주파수

Wsc=Wc/5;               % 속도제어계의 교차각주파수

Wpi=Wsc/5;              % PI제어기의 절점주파수
voltage_max=75.0;       % 모터 정격전압
current_max=18.8;       % 모터 최대전류

% Sampling Period
=============================================

T_t=1.5;                % 전체 실행 시간
T_s=0.001;              % 속도제어계 샘플링타임
T_a=0.0001;             % 가속도제어계 샘플링타임
dt=0.000001;            % 모터 시뮬레이션

T_1=T_t/T_s;
T_2=T_s/T_a;
T_3=T_a/dt;
count=0;

% Motor Parameters
=================================================

Ra_1=1.02;
La_1=0.00107;
Jm_1=0.000245;
Dm_1=0.00105;
Kt_1=0.22246;
Ke_1=0.2333;

Ra_2=1.02;
La_2=0.00107;
Jm_2=0.000245;
Dm_2=0.00105;
Kt_2=0.22246;
Ke_2=0.2333;

Ra_3=1.53;
La_3=0.00175;
Jm_3=0.000176;
Dm_3=0.00145;
Kt_3=0.2156;
Ke_3=0.2262;

Ra_4=1.53;
La_4=0.00175;
Jm_4=0.000176;
Dm_4=0.00145;
Kt_4=0.2156;
Ke_4=0.2262;

% Controllers Gain
=================================================

Kap_1=Wc*La_1;          % 제어기 게인 설정
Kai_1=(Wc*Ra_1);
Ksp_1=Jm_1*Wsc/Kt_1;
Ksi_1=(Ksp_1*Wpi);
Kpp_1=1;

Kap_2=Wc*La_2;
Kai_2=(Wc*Ra_2);
Ksp_2=Jm_2*Wsc/Kt_2;
Ksi_2=(Ksp_2*Wpi);
Kpp_2=1;

Kap_3=Wc*La_3;
Kai_3=(Wc*Ra_3);
Ksp_3=Jm_3*Wsc/Kt_3;
Ksi_3=(Ksp_3*Wpi);
Kpp_3=1;

Kap_4=Wc*La_4;
Kai_4=(Wc*Ra_4);
Ksp_4=Jm_4*Wsc/Kt_4;
Ksi_4=(Ksp_4*Wpi);
Kpp_4=1;

% System Initializer
=============================================

w_rpm=600;
w_ref=0;
w_rad=w_rpm*(2*pi)/60;
sync_t=0;
dummy1=0;
dummy2=0;
dummy3=0;
big=0;
sync_er=0;
sync_u=0;

a_1=0;
Ga_1=0;
Ga_1_o=0;
dw_1=0;
dw_1_o=0;
```

```
w_1=0;
w_1_rpm=w_1*60/(2*pi);
w_1_o=0;
e_a_1=0;
e_a_1_o=0;
e_w_1=0;
e_w_1_o=0;
i_p_1=0;
i_i_1=0;
i_ref_1=0;
Ia_1=0;
e_i_1=0;
e_i_1_o=0;
v_p_1=0;
v_i_1=0;
v_a_1=0;
v_a_1_o=0;
v_t_1=0;
v_e_1=0;
v_ref_1=0;
Td_1=0;
sync_1=0;
sync_1_o=0;
sync_1_er=0;
sync_1_er_o=0;
sync_1_u=0;
a_ref_1=0;
a_ref_1_o=0;
a_2=0;
Ga_2=0;
Ga_2_o=0;
dw_2=0;
dw_2_o=0;
e_a_2=0;
e_a_2_o=0;
w_2=0;
w_2_rpm=w_2*60/(2*pi);
w_2_o=0;
e_w_2=0;
e_w_2_o=0;
i_p_2=0;
i_i_2=0;
i_ref_2=0;
Ia_2=0;
e_i_2=0;
e_i_2_o=0;
v_p_2=0;
v_i_2=0;
v_a_2=0;
v_a_2_o=0;
v_t_2=0;
v_e_2=0;
v_ref_2=0;
Td_2=0;
sync_2=0;
sync_2_o=0;
sync_2_er=0;
sync_2_er_o=0;
sync_2_u=0;
a_ref_2=0;
a_ref_2_o=0;
a_3=0;
Ga_3=0;
Ga_3_o=0;
dw_3=0;
dw_3_o=0;
w_3=0;
w_3_rpm=w_1*60/(2*pi);
e_a_3=0;
e_a_3_o=0;
w_3_o=0;
e_w_3=0;
e_w_3_o=0;
i_p_3=0;
i_i_3=0;
i_ref_3=0;
Ia_3=0;
e_i_3=0;
e_i_3_o=0;
v_p_3=0;
v_i_3=0;
v_a_3=0;
v_a_3_o=0;
v_t_3=0;
v_e_3=0;
v_ref_3=0;
Td_3=0;
sync_3=0;
sync_3_o=0;
sync_3_er=0;
sync_3_er_o=0;
sync_3_u=0;
a_ref_3=0;
a_ref_3_o=0;
a_4=0;
Ga_4=0;
Ga_4_o=0;
dw_4=0;
dw_4_o=0;
w_4=0;
w_4_rpm=w_2*60/(2*pi);
w_4_o=0;
e_a_4=0;
e_a_4_o=0;
e_w_4=0;
e_w_4_o=0;
i_p_4=0;
i_i_4=0;
i_ref_4=0;
Ia_4=0;
e_i_4=0;
e_i_4_o=0;
v_p_4=0;
v_i_4=0;
v_a_4=0;
```

```
v_a_4_o=0;
v_t_4=0;
v_e_4=0;
v_ref_4=0;

Td_4=0;

sync_4=0;

sync_4_o=0;

sync_4_er=0;

sync_4_er_o=0;

sync_4_u=0;
a_ref_4=0;
a_ref_4_o=0;

% Cycle and Time Shearing Initialize
==============================

T_0=T_1+1;
y0=zeros(T_0,1);        %time vector
y1=y0;                  %No.1 speed
y2=y0;                  %No.2 speed
y3=y0;                  %No.3 speed
y4=y0;                  %No.4 speed
y5=y0;                  %No.1 current
y6=y0;                  %No.2 current
y7=y0;                  %No.3 current
y8=y0;                  %No.4 current
y9=y0;                  %No.1 voltage
y10=y0;                 %No.2 voltage
y11=y0;                 %No.3 voltage
y12=y0;                 %No.4 voltage
y13=y0;                 %No.1 synchronous error
y14=y0;                 %No.2 synchronous error
y15=y0;                 %No.3 synchronous error
y16=y0;                 %No.4 synchronous error

y0=0:T_s:T_t;
y1(1,1)=w_1_rpm;
y2(1,1)=w_2_rpm;
y3(1,1)=w_3_rpm;
y4(1,1)=w_4_rpm;

y5(1,1)=Ia_1;

y6(1,1)=Ia_2;

y7(1,1)=Ia_3;

y8(1,1)=Ia_4;
y9(1,1)=v_t_1;
y10(1,1)=v_t_2;
y11(1,1)=v_t_3;
y12(1,1)=v_t_4;

y13(1,1)=sync_1_er;

y14(1,1)=sync_2_er;

y15(1,1)=sync_3_er;

y16(1,1)=sync_4_er;

% Speed Loop Start
================================================

for k_v=1:1:T_1

 if k_v <= (1/T_s)                  %Speed Refrence
     w_ref=w_ref+w_rad/(1/T_s);
%  elseif k_v > (0.6/T_s)
%       w_ref=w_ref-w_rad/(0.6/T_s);
 else
     w_ref=w_rad;
 end

 if k_v <= (1.2/T_s)                %Disturbance
     Td_1=0;

     Td_2=0;

     Td_3=0;

     Td_4=0;
  else
     Td_4=0.47726; % 정격 50% 외란입력
 end

e_w_1=w_ref-w_1-sync_1_u; % 1axis speed error
a_ref_1=a_ref_1_o+(Ksp_1*e_w_1)+(Ksi_1*T_s*e_w_1)-(Ksp_1
*e_w_1_o); % 가속도 지령

e_w_2=w_ref-w_2-sync_2_u; % 2axis speed error
a_ref_2=a_ref_2_o+(Ksp_2*e_w_2)+(Ksi_2*T_s*e_w_2)-(Ksp_2
*e_w_2_o); % 가속도 지령

e_w_3=w_ref-w_3-sync_3_u; % 1axis speed error
a_ref_3=a_ref_3_o+(Ksp_3*e_w_3)+(Ksi_3*T_s*e_w_3)-(Ksp_3
*e_w_3_o); % 가속도 지령

e_w_4=w_ref-w_4-sync_4_u; % 2axis speed error
a_ref_4=a_ref_4_o+(Ksp_4*e_w_4)+(Ksi_4*T_s*e_w_4)-(Ksp_4
*e_w_4_o); % 가속도 지령

e_w_1_o=e_w_1;
e_w_2_o=e_w_2;
e_w_3_o=e_w_3;
e_w_4_o=e_w_4;
a_ref_1_o=a_ref_1;
a_ref_2_o=a_ref_2;
a_ref_3_o=a_ref_3;
a_ref_4_o=a_ref_4;


% Acceleration Loop Start
==========================================

for k_i=1:1:T_2;

e_a_1=a_ref_1-a_1; % 1축 가속도 오차
a_1=(Jm_1/Kt_1)*Ga_1;  % 가속도 피드백
v_a_1=v_a_1_o+(Kcp_1*e_a_1)+(Kci_1*T_i*e_a_1)-(Kcp_1*e_a
_1_o); % 전압지령
v_t_1=v_a_1;
v_e_1=Ke_1*w_1;
v_ref_1=v_t_1-v_e_1;

e_a_2=a_ref_2-a_2; % 2축 가속도 오차
a_2=(Jm_2/Kt_2)*Ga_2;  % 가속도 피드백
v_a_2=v_a_2_o+(Kcp_2*e_a_2)+(Kci_2*T_i*e_a_2)-(Kcp_2*e_a
_2_o); % 전압지령
```

```
v_t_2=v_a_2;
v_e_2=Ke_2*w_2;
v_ref_2=v_t_2-v_e_2;

e_a_3=a_ref_3-a_3;   % 3축 가속도 오차
a_3=(Jm_3/Kt_3)*Ga_3;  % 가속도 피드백
v_a_3=v_a_3_o+(Kcp_3*e_a_3)+(Kci_3*T_i*e_a_3)-(Kcp_3*e_a
_3_o); % 전압지령
v_t_3=v_a_3;
v_e_3=Ke_3*w_3;
v_ref_3=v_t_3-v_e_3;

e_a_4=a_ref_4-a_4;   % 4축 가속도 오차
a_4=(Jm_4/Kt_4)*Ga_4;  % 가속도 피드백
v_a_4=v_a_4_o+(Kcp_4*e_a_4)+(Kci_4*T_i*e_a_4)-(Kcp_4*e_a
_4_o); % 전압지령
v_t_4=v_a_4;
v_e_4=Ke_4*w_4;
v_ref_4=v_t_4-v_e_4;

v_a_1_o=v_a_1;
v_a_2_o=v_a_2;
v_a_3_o=v_a_3;
v_a_4_o=v_a_4;

e_a_1_o=e_a_1;
e_a_2_o=e_a_2;
e_a_3_o=e_a_3;
e_a_4_o=e_a_4;

% Motor Model Simulation Start
=====================================

for k_m=1:1:T_3;

dI_1= -(Ra_1/La_1)*Ia_1+(1/La_1)*v_ref_1;
Ia_1=Ia_1+dI_1*dt;

dI_2= -(Ra_2/La_2)*Ia_2+(1/La_2)*v_ref_2;
Ia_2=Ia_2+dI_2*dt;

dI_3= -(Ra_3/La_3)*Ia_3+(1/La_3)*v_ref_3;
Ia_3=Ia_3+dI_3*dt;

dI_4= -(Ra_4/La_4)*Ia_4+(1/La_4)*v_ref_4;
Ia_4=Ia_4+dI_4*dt;

end

% Motor Model Simulation End
=====================================

Te_1=Kt_1*Ia_1;
Tm_1=Te_1-Td_1;

Te_2=Kt_2*Ia_2;
Tm_2=Te_2-Td_2;

Te_3=Kt_3*Ia_3;
Tm_3=Te_3-Td_3;

Te_4=Kt_4*Ia_4;
Tm_4=Te_4-Td_4;

% Each axis acceleration filtering
=====================================
dw_1=(Tm_1-Dm_1*w_1)/Jm_1;
Ga_1=0.6321*dw_1_o + 0.3679*Ga_1_o;

% 2axis acceleration filtering
dw_2=(Tm_2-Dm_2*w_2)/Jm_2;
Ga_2=0.6321*dw_2_o + 0.3679*Ga_2_o;

% 3axis acceleration filtering
dw_3=(Tm_3-Dm_3*w_3)/Jm_3;
Ga_3=0.6321*dw_3_o + 0.3679*Ga_3_o;

% 4axis acceleration filtering
dw_4=(Tm_4-Dm_4*w_4)/Jm_4;
Ga_4=0.6321*dw_4_o + 0.3679*Ga_4_o;

dw_1_o=dw_1;
dw_2_o=dw_2;
dw_3_o=dw_3;
dw_4_o=dw_4;
Ga_1_o=Ga_1;
Ga_2_o=Ga_2;
Ga_3_o=Ga_3;
Ga_4_o=Ga_4;

end

% Acceleration Loop End
=============================================

w_1=w_1_o+dw_1*T_s;    % 속도연산

w_2=w_2_o+dw_2*T_s;

w_3=w_3_o+dw_3*T_s;

w_4=w_4_o+dw_4*T_s;

sync_1=sync_1_o+(w_1-w_1_o)*T_s;    % 샘플링당 변경된
위치정보 연산
sync_2=sync_2_o+(w_2-w_2_o)*T_s;
sync_3=sync_3_o+(w_3-w_3_o)*T_s;
sync_4=sync_4_o+(w_4-w_4_o)*T_s;

% Max position synchronous error comparison
```

```
com1_1=sync_1-sync_2;

com1_2=sync_1-sync_3;

com1_3=sync_1-sync_4;

dummy1_1=abs(dum1_1);

dummy1_2=abs(dum1_2);

dummy1_3=abs(dum1_3);
m1=[dummy1_1 dummy1_2 dummy1_3];
big1=max(m1);
if big1==dummy1_1

    sync_1_er=sync_1-sync_2;

elseif big1==dummy1_2

    sync_1_er=sync_1-sync_3;
else big1==dummy1_3

    sync_1_er=sync_1-sync_4;
end


com2_1=sync_2-sync_3;

com2_2=sync_2-sync_4;

com2_3=sync_2-sync_1;

dummy2_1=abs(dum2_1);

dummy2_2=abs(dum2_2);

dummy2_3=abs(dum2_3);
m2=[dummy2_1 dummy2_2 dummy2_3];
big2=max(m2);
if big2==dummy2_1

    sync_2_er=sync_2-sync_3;

elseif big2==dummy2_2

    sync_2_er=sync_2-sync_4;
else big2==dummy2_3

    sync_2_er=sync_2-sync_1;
end


com3_1=sync_3-sync_4;

com3_2=sync_3-sync_1;

com3_3=sync_3-sync_2;

dummy3_1=abs(dum3_1);

dummy3_2=abs(dum3_2);

dummy3_3=abs(dum3_3);
m3=[dummy3_1 dummy3_2 dummy3_3];
big3=max(m3);
if big3==dummy3_1

    sync_3_er=sync_3-sync_4;

elseif big3==dummy3_2

    sync_3_er=sync_3-sync_1;
else big3==dummy3_3

    sync_3_er=sync_3-sync_2;
end


com4_1=sync_4-sync_1;

com4_2=sync_4-sync_2;

com4_3=sync_4-sync_3;

dummy4_1=abs(dum4_1);

dummy4_2=abs(dum4_2);

dummy4_3=abs(dum4_3);
m4=[dummy4_1 dummy4_2 dummy4_3];
big4=max(m4);
if big4==dummy4_1

    sync_4_er=sync_4-sync_1;

elseif big4==dummy4_2

    sync_4_er=sync_4-sync_2;
else big4==dummy4_3

    sync_4_er=sync_4-sync_3;
end


sync_1_u=Kpp_1*sync_1_er;

sync_2_u=Kpp_2*sync_2_er;

sync_3_u=Kpp_3*sync_3_er;

sync_4_u=Kpp_4*sync_4_er;


w_1_o=w_1;
w_2_o=w_2;
w_3_o=w_3;
w_4_o=w_4;


sync_1_o=sync_1;

sync_2_o=sync_2;

sync_3_o=sync_3;

sync_4_o=sync_4;



sync_1_er_o=sync_1_er;

sync_2_er_o=sync_2_er;

sync_3_er_o=sync_3_er;

sync_4_er_o=sync_4_er;


w_1_rpm=w_1*60/(2*pi);
w_2_rpm=w_2*60/(2*pi);
w_3_rpm=w_3*60/(2*pi)
w_4_rpm=w_4*60/(2*pi);
y1(k_v+1,1)=w_1_rpm;
y2(k_v+1,1)=w_2_rpm;
y3(k_v+1,1)=w_3_rpm;
y4(k_v+1,1)=w_4_rpm;

y5(k_v+1,1)=Ia_1;

y6(k_v+1,1)=Ia_2;

y7(k_v+1,1)=Ia_3;

y8(k_v+1,1)=Ia_4;

y9(k_v+1,1)=v_t_1;
y10(k_v+1,1)=v_t_2;
y11(k_v+1,1)=v_t_3;
y12(k_v+1,1)=v_t_4;

y13(k_v+1,1)=sync_1_er;
```

```
y14(k_v+1,1)=sync_2_er;
y15(k_v+1,1)=sync_3_er;
y16(k_v+1,1)=sync_4_er;

count=count+1;

end

% Speed Loop End
=============================================

figure(1)
set (gcf, 'NumberTitle','off')
set (gcf, 'Name','Speed')
plot(y0,y1,y0,y2,y0,y3,y0,y4)
xlabel('time[sec]')
ylabel('speed[rpm]')
grid

figure(2)
set (gcf, 'NumberTitle','off')
set (gcf, 'Name','Current')
plot(y0,y5,y0,y6,y0,y7,y0,y8)
xlabel('time[sec]')
ylabel('current[A]')
grid

figure(3)
set (gcf, 'NumberTitle','off')
set (gcf, 'Name','Voltage')
plot(y0,y9,y0,y10,y0,y11,y0,y12)
xlabel('time[sec]')
ylabel('voltage[V]')
grid

figure(4)
set (gcf, 'NumberTitle','off')
set (gcf, 'Name','Position synchronous error')
plot(y0,y13,y0,y14,y0,y15,y0,y16)
xlabel('time[sec]')
ylabel('Position synchronous error[rad]')
grid
```

A-5 4축 위치동기제어 실기실험 프로그램

```
/*************************************************/
/* Position Synchronizing Control System         */
/*          Realtime Monitoring Program          */
/*   Speed, Current, Disturbance,                */
     Position synchronous error                  */
/*                                               */
/*              by Choi bong-seok. 2003. 12.     */
/*************************************************/

#include <bios.h>
#include <dos.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#define     PORT        0x300
#define     B_ADR       0x120
#define     XYCH        B_ADR+0x000
#define     ZUCH        B_ADR+0x001
#define TMR_CMD         B_ADR+0x01A
#define TMR_DAT         B_ADR+0x018
#define IN_PORT         B_ADR+0x01C
#define OUT_PORT B_ADR+0x01C
#define TMR_RST         B_ADR+0x01D
#define OP_PORT B_ADR+0x01E
#define CLTCH_RST B_ADR+0x01F
#define pi   3.14159265358979
#define data_sum 500
#define T_i 0.0005
#define T_s 0.002

/*******************************************************/
#define kap1 0.32
#define kai1 0.32
#define ksp1 0.13
#define ksi1 -0.12
#define kpp1 1
#define kap2 0.31
#define kai2 0.33
#define ksp2 0.13
#define ksi2 -0.12
#define kpp2 1
#define kap3 0.43
#define kai3 0.31
#define ksp3 0.13
#define ksi3 -0.12
#define kpp3 1
#define kap4 0.51
#define kai4 0.23
#define ksp4 0.13
#define ksi4 -0.12
#define kpp4 1
#define Kt1 0.22246
#define Jm1 0.000245
#define K1 0.0012  /* 0.000863265  */
#define Kt2 0.22246
#define Jm2 0.000245
#define K2 0.0012  /* 0.000863265  */
#define Kt3 0.2156
#define Jm3 0.000176
#define K3 0.0012  /* 0.000863265  */
#define Kt4 0.2156
#define Jm4 0.000176
#define K4 0.0002  /* 0.000863265  */
/*******************************************************/
int DA, i, con, cw_flg_1, cw_flg_2, cw_flg_3, cw_flg_4,
cw_flg, ct;
int acceleration loop_flag, speedloop_flag, select_flag,
ref_flag, skip_flag;
int DA_1, DA_1_h, DA_1_l, AD_1_h, AD_1_l;
int DA_2, DA_2_h, DA_2_l, AD_2_h, AD_2_l;
int DA_3, DA_3_h, DA_3_l, AD_3_h, AD_3_l;
int DA_4, DA_4_h, DA_4_l, AD_4_h, AD_4_l;
float ref, ref_rpm, ref_max, ref_speed;
float ref_step, time_sum, gap[2], x_gap, up_slop,
down_slop;
float DA_1_scale, AD_1, current_1[2], voltage_1,
rpm_1,cur1[2];
float DA_2_scale, AD_2, current_2[2], voltage_2,
rpm_2,cur2[2];
float DA_3_scale, AD_3, current_3[2], voltage_3,
rpm_3,cur3[2];
float DA_4_scale, AD_4, current_4[2], voltage_4,
rpm_4,cur4[2];
long v;
long pulse_1[2], count_1, count_1_gap, xd3, xd2, xd1, xd0;
long pulse_2[2], count_2, count_2_gap, yd3, yd2, yd1, yd0;
long pulse_3[2], count_3, count_3_gap, zd3, zd2, zd1, zd0;
long pulse_4[2], count_4, count_4_gap, ud3, ud2, ud1, ud0;
float speed_1[2], theta_1[2], w_1_e[2], A_1[2], A_1_e[2],
V_1[2];
float speed_2[2], theta_2[2], w_2_e[2], A_2[2], A_2_e[2],
V_2[2];
float speed_3[2], theta_3[2], w_3_e[2], A_3[2], A_3_e[2],
V_3[2];
float speed_4[2], theta_4[2], w_4_e[2], A_4[2], A_4_e[2],
```

```
V_4[2];
float speed_e_1, speed_e_2, speed_e_3, speed_e_4;
float theta_e, theta_e_1, theta_e_2, theta_e_3, theta_e_4;
float theta_ref, theta_ref_1, theta_ref_2, theta_ref_3,
theta_ref_4, skip;
float data_spd1[501], data_spd2[501], data_spd3[501],
data_spd4[501];
float data_cur1[501], data_cur2[501], data_cur3[501],
data_cur4[501];
float data_tht1[501], data_tht2[501], data_tht3[501],
data_tht4[501];
float dummy1, dummy2, dummy3;
float gasok_1[2],gasok_1ob[2],gasok_2[2],gasok_2ob[2];
float gasok_3[2],gasok_3ob[2],gasok_4[2],gasok_4ob[2];
float y;
int MaxX, MaxY, MaxColors, ErrorCode;
int x_step, y_step, x1, y1, x2, y2, y3, xx1[2], yy1[2],
yy2[2], yy3[2], yy4[2], yy5[2];
int gprintf( int xloc, int yloc, char *fmt, ... );
struct time start, end;
struct palettetype palette;
struct viewporttype vp;
struct viewporttype vp1;
FILE *fp;
void initial_time(void);
void initial_counter(void);
void initial_data(void);
void initial_graph(void);
void realtime_monitoring(void);
void data_saving(void);
void monitor_set(void);
void monitor_graphset1(void);
void monitor_graphset2(void);
void monitor_graphset3(void);
void realtime1(void);
void realtime2(void);
void realtime3(void);
void realtime4(void);
void realtime5(void);
void realtime1_grap(void);
void realtime2_grap(void);
void realtime3_grap(void);
void realtime4_grap(void);
void realtime5_grap(void);
void ramp_reference(void);
void variable_reference(void);
void data_grap(void);
void data_out(void);
void selection1(void);
void selection2(void);
void selection3(void);
void Fillbox(int sx, int sy, int ex, int ey, int color);
void timer(void);
void interrupt(*oldfunc)();
void interrupt current(void);
void control_input_start_end(void);
void speed(void);
void ad_con(double *value);
void reference1(void);
void reference2(void);
void reference3(void);
/*********************************************************/
/********************  Main Program  *********************/
/*********************************************************/
void interrupt acceleration(void){
acceleration loop_flag++;
v++;
if(v<10000)
outportb(PORT+3, 0x0F);
else if(v<20000)
outportb(PORT+3, 0x1F);
else
outportb(PORT+3, 0x0F);
outportb(PORT+2, 0x00);
outportb(PORT+0, 0x00);
do{
con = inportb(PORT+8);
                }while(con>127);
 AD_1_l = inportb(PORT+0);
 AD_1_h = inportb(PORT+1);      // current A/D converting
 outportb(PORT+2, 0x11);
 outportb(PORT+0, 0x00);
 do{
 con = inportb(PORT+8);
 }while(con>127);
 AD_2_l = inportb(PORT+0);
 AD_2_h = inportb(PORT+1);
outportb(PORT+2, 0x44);
outportb(PORT+0, 0x00);
do{
con = inportb(PORT+8);
 }while(con>127);
AD_3_l = inportb(PORT+0);
AD_3_h = inportb(PORT+1);
outportb(PORT+2, 0x55);
outportb(PORT+0, 0x00);
do{
con = inportb(PORT+8);
}while(con>127);
AD_4_l = inportb(PORT+0);
AD_4_h = inportb(PORT+1);
/*********************************************************/
AD_1 = (float)((AD_1_h<<4)+(AD_1_l>>4));
current_1[1] = 5.*((AD_1-2047.)/2048.)+0.3;
// current calculation
cur1[1] = 0.1339*current_1[0] + 0.8661*cur1[0];
// current filtering
AD_2 = (float)((AD_2_h<<4)+(AD_2_l>>4));
current_2[1] = 5.*((AD_2-2047.)/2048.)+0.3;
cur2[1] = 0.1339*current_2[0] + 0.8661*cur2[0];
```

```
AD_3 = (float)((AD_3_h<<4)+(AD_3_l>>4));
current_3[1] = 5.*((AD_3-2047.)/2048.)+0.3;
cur3[1] = 0.1479*current_3[0] + 0.8521*cur3[0];
AD_4 = (float)((AD_4_h<<4)+(AD_4_l>>4));
current_4[1] = 5.*((AD_4-2047.)/2048.)+0.3;
cur4[1] = 0.1479*current_4[0] + 0.8521*cur4[0];
outportb(OP_PORT, 0x40);
 outportb(XYCH+10, 0x01);
xd3 = inportb(XYCH+6);
xd2 = inportb(XYCH+4);
xd1 = inportb(XYCH+2);
xd0 = inportb(XYCH+0);
pulse_1[1] = ((xd3<<24)+(xd2<<16)+(xd1<<8)+xd0);
// count board initialize
outportb(XYCH+10, 0x02);
yd3 = inportb(XYCH+6);
yd2 = inportb(XYCH+4);
yd1 = inportb(XYCH+2);
yd0 = inportb(XYCH+0);
pulse_2[1] = ((yd3<<24)+(yd2<<16)+(yd1<<8)+yd0);
outportb(ZUCH+10, 0x01);
zd3 = inportb(ZUCH+6);
zd2 = inportb(ZUCH+4);
zd1 = inportb(ZUCH+2);
zd0 = inportb(ZUCH+0);
pulse_3[1] = ((zd3<<24)+(zd2<<16)+(zd1<<8)+zd0);
outportb(ZUCH+10, 0x02);
ud3 = inportb(ZUCH+6);
ud2 = inportb(ZUCH+4);
ud1 = inportb(ZUCH+2);
ud0 = inportb(ZUCH+0);
pulse_4[1] = ((ud3<<24)+(ud2<<16)+(ud1<<8)+ud0);
outportb(OP_PORT, 0x00);
outportb(CLTCH_RST, 0x00);
count_1_gap = (pulse_1[1]-pulse_1[0]);
count_1 = (count_1_gap < 0) ? count_1_gap+2147483647. :
count_1_gap;
speed_1[1] = (((float)(count_1)*2.*pi)/4000.)/T_i;
speed_1[1] = (speed_1[1] + speed_1[0])/2.;
rpm_1 = (speed_1[1]*(60./(2.*pi)));
// motor rpm caculation
count_2_gap = (pulse_2[1]-pulse_2[0]);
count_2 = (count_2_gap < 0) ? count_2_gap+2147483647. :
count_2_gap;
speed_2[1] = (((float)(count_2)*2.*pi)/4000.)/T_i;
speed_2[1] = (speed_2[1] + speed_2[0])/2.;
rpm_2 = (speed_2[1]*(60./(2.*pi)));
count_3_gap = (pulse_3[1]-pulse_3[0]);
count_3 = (count_3_gap < 0) ? count_3_gap+2147483647. :
count_3_gap;
speed_3[1] = (((float)(count_3)*2.*pi)/4000.)/T_i;
speed_3[1] = (speed_3[1] + speed_3[0])/2.;
rpm_3 = (speed_3[1]*(60./(2.*pi)));
count_4_gap = (pulse_4[1]-pulse_4[0]);
count_4 = (count_4_gap < 0) ? count_4_gap+2147483647. :
count_4_gap;
speed_4[1] = (((float)(count_4)*2.*pi)/4000.)/T_i;
speed_4[1] = (speed_4[1] + speed_4[0])/2.;
    rpm_4 = (speed_4[1]*(60./(2.*pi)));
    gasok_1[1] = (speed_1[1] - speed_1[0])/T_i;  //
acceleration caculation
    gasok_1ob[1] = (0.1832*gasok_1[0] +
    0.8168*gasok_1ob[0]);  // acceleration filtering
    gasok_2[1] = (speed_2[1] - speed_2[0])/T_i;
    gasok_2ob[1]      =      (0.1832*gasok_2[0]      +
0.8168*gasok_2ob[0]);
    gasok_3[1] = (speed_3[1] - speed_3[0])/T_i;
    gasok_3ob[1]      =      (0.1832*gasok_3[0]      +
0.8168*gasok_3ob[0]);
    gasok_4[1] = (speed_4[1] - speed_4[0])/T_i;
    gasok_4ob[1]      =      (0.1832*gasok_4[0]      +
0.8168*gasok_4ob[0]);
    theta_1[1] = theta_1[0]+(speed_1[1]-speed_1[0])*T_i;
// motor position variation
    theta_2[1] = theta_2[0]+(speed_2[1]-speed_2[0])*T_i;
    theta_3[1] = theta_3[0]+(speed_3[1]-speed_3[0])*T_i;
    theta_4[1] = theta_4[0]+(speed_4[1]-speed_4[0])*T_i;

/******************** max position error comparition
*************************/
    dummy1 = fabs(theta_1[1]-theta_2[1]);
    dummy2 = fabs(theta_1[1]-theta_3[1]);
    dummy3 = fabs(theta_1[1]-theta_4[1]);
    if(dummy1>dummy2)
      if(dummy1>dummy3)
        theta_e_1 = theta_1[1]-theta_2[1];
      else
        theta_e_1 = theta_1[1]-theta_4[1];
    else
      if(dummy2>dummy3)
        theta_e_1 = theta_1[1]-theta_3[1];
      else
        theta_e_1 = theta_1[1]-theta_4[1];
    dummy1 = fabs(theta_2[1]-theta_3[1]);
    dummy2 = fabs(theta_2[1]-theta_4[1]);
    dummy3 = fabs(theta_2[1]-theta_1[1]);
    if(dummy1>dummy2)
      if(dummy1>dummy3)
        theta_e_2 = theta_2[1]-theta_3[1];
      else
        theta_e_2 = theta_2[1]-theta_1[1];
    else
      if(dummy2>dummy3)
        theta_e_2 = theta_2[1]-theta_4[1];
      else
        theta_e_2 = theta_2[1]-theta_1[1];
    dummy1 = fabs(theta_3[1]-theta_4[1]);
    dummy2 = fabs(theta_3[1]-theta_1[1]);
    dummy3 = fabs(theta_3[1]-theta_2[1]);
    if(dummy1>dummy2)
      if(dummy1>dummy3)
```

```
        theta_e_3 = theta_3[1]-theta_4[1];
      else
        theta_e_3 = theta_3[1]-theta_2[1];
    else
      if(dummy2>dummy3)
        theta_e_3 = theta_3[1]-theta_1[1];
else
theta_e_3 = theta_3[1]-theta_2[1];
dummy1 = fabs(theta_4[1]-theta_1[1]);
dummy2 = fabs(theta_4[1]-theta_2[1]);
dummy3 = fabs(theta_4[1]-theta_3[1]);
if(dummy1>dummy2)
if(dummy1>dummy3)
theta_e_4 = theta_4[1]-theta_1[1];
else
theta_e_4 = theta_4[1]-theta_3[1];
else
if(dummy2>dummy3)
theta_e_4 = theta_4[1]-theta_2[1];
else
theta_e_4 = theta_4[1]-theta_3[1];
theta_ref_1 = kpp1*theta_e_1;
// feedback synchronous error value
theta_ref_2 = kpp2*theta_e_2;
theta_ref_3 = kpp3*theta_e_3;
theta_ref_4 = kpp4*theta_e_4;

/*****************************************************/
 while(acceleration loop_flag==skip_flag) speed();
 if(ref_flag==0)
 reference1();
 else if(ref_flag==1)
 reference2();
 else
 reference3();
 ref_speed = (2.*pi*ref_rpm)/60.;
 ref_max = (2.*pi*ref)/60.;
/*****************************************************/
 A_1_e[1] = A_1[1]-gasok_1ob[1]*K1;
 // acceleration error
 V_1[1] = V_1[0]+kap1*A_1_e[1]+kai1*A_1_e[0];
 // voltage ref.
 if(V_1[1] >= 0)
 cw_flg_1 = 1;
 else
 cw_flg_1 = 0;
 if(V_1[1] >= 0)
 voltage_1 = V_1[1];
 else
 voltage_1 = -V_1[1];
 DA_1_scale = voltage_1*(4095./30.);
 DA_1_scale = (DA_1_scale >= 4095.) ? 4095. :
 (DA_1_scale <= 0) ? 0 : DA_1_scale;
 DA_1 = (int)(DA_1_scale);
 DA_1_l = (DA_1 & 0x00ff) << 4;
 DA_1_h = (DA_1 & 0x0ff0) >> 4;
 // analog pwm rate ref.
 A_2_e[1] = A_2[1]-gasok_2ob[1]*K2;
 V_2[1] = V_2[0]+kap2*A_2_e[1]+kai2*A_2_e[0];
 if(V_2[1] >= 0)
 cw_flg_2 = 1;
 else
 cw_flg_2 = 0;
 if(V_2[1] >= 0)
   voltage_2 = V_2[1];
   else
   voltage_2 = -V_2[1];
   DA_2_scale = voltage_2*(255./30.);
   DA_2_scale = (DA_2_scale >= 255.) ? 255. :
   (DA_2_scale <= 0) ? 0 : DA_2_scale;
   DA_2 = (int)(DA_2_scale);
   A_3_e[1] = A_3[1]-gasok_3ob[1]*K3;
   V_3[1] = V_3[0]+kap3*A_3_e[1]+kai3*A_3_e[0];
   if(V_3[1] >= 0)
     cw_flg_3 = 1;
   else
     cw_flg_3 = 0;
   if(V_3[1] >= 0)
     voltage_3 = V_3[1];
   else
     voltage_3 = -V_3[1];
   DA_3_scale = voltage_3*(4095./30.);
   DA_3_scale = (DA_3_scale >= 4095.) ? 4095. :
   (DA_3_scale <= 0) ? 0 : DA_3_scale;
   DA_3 = (int)(DA_3_scale);
   DA_3_l = (DA_3 & 0x00ff) << 4;
   DA_3_h = (DA_3 & 0x0ff0) >> 4;
   A_4_e[1] = A_4[1]-gasok_4ob[1]*K4;
   V_4[1] = V_4[0]+kap4*A_4_e[1]+kai4*A_4_e[0];
   if(V_4[1] >= 0)
     cw_flg_4 = 1;
   else
     cw_flg_4 = 0;
   if(V_4[1] >= 0)
     voltage_4 = V_4[1];
   else
    voltage_4 = -V_4[1];
   DA_4_scale = voltage_4*(255./30.);
   DA_4_scale = (DA_4_scale >= 255.) ? 255. :
   (DA_4_scale <= 0) ? 0 : DA_4_scale;
   DA_4 = (int)(DA_4_scale);
   cw_flg =15;
   outportb(PORT+4, DA_1_l);
   outportb(PORT+5, DA_1_h);
   outportb(PORT+10, DA_2);
   outportb(PORT+6, DA_3_l);
   outportb(PORT+7, DA_3_h);
   outportb(PORT+11, DA_4);
   // analog D/A output
   /*****************************************************/
   gasok_1[0]=gasok_1[1];
   gasok_1ob[0]=gasok_1ob[1];
   pulse_1[0] = pulse_1[1];
   speed_1[0] = speed_1[1];
   theta_1[0] = theta_1[1];
   A_1_e[0] = A_1_e[1];
```

```
    V_1[0] = V_1[1];
    current_1[0] = current_1[1];
    cur1[0] = cur1[1];

    gasok_2[0]=gasok_2[1];

gasok_2ob[0]=gasok_2ob[1];
pulse_2[0] = pulse_2[1];
speed_2[0] = speed_2[1];
theta_2[0] = theta_2[1];
A_2_e[0] = A_2_e[1];
V_2[0] = V_2[1];
current_2[0] = current_2[1];
cur2[0] = cur2[1];

gasok_3[0]=gasok_3[1];

gasok_3ob[0]=gasok_3ob[1];
pulse_3[0] = pulse_3[1];
speed_3[0] = speed_3[1];
theta_3[0] = theta_3[1];
A_3_e[0] = A_3_e[1];
V_3[0] = V_3[1];
current_3[0] = current_3[1];
cur3[0] = cur3[1];

gasok_4[0]=gasok_4[1];

gasok_4ob[0]=gasok_4ob[1];
pulse_4[0] = pulse_4[1];
speed_4[0] = speed_4[1];
theta_4[0] = theta_4[1];
A_4_e[0] = A_4_e[1];
V_4[0] = V_4[1];
current_4[0] = current_4[1];
cur4[0] = cur4[1];

outportb(PORT+8, 0x10);

outportb(0x20, 0x20);
}
/*************************************************************/
/******************** Speed Routine **********************/
/*************************************************************/
void speed(void){
w_1_e[1] = ref_speed-speed_1[1]-theta_ref_1;
// speed controller input ref.
A_1[1] = A_1[0]+ksp1*w_1_e[1]+ksi1*w_1_e[0];

// acceleration ref.
w_2_e[1] = ref_speed-speed_2[1]-theta_ref_2;

A_2[1] = A_2[0]+ksp2*w_2_e[1]+ksi2*w_2_e[0];
w_3_e[1] = ref_speed-speed_3[1]-theta_ref_3;

A_3[1] = A_3[0]+ksp3*w_3_e[1]+ksi3*w_3_e[0];

w_4_e[1] = ref_speed-speed_4[1]-theta_ref_4;

A_4[1] = A_4[0]+ksp4*w_4_e[1]+ksi4*w_4_e[0];
/*************************************************************/
w_1_e[0] = w_1_e[1];
A_1[0] = A_1[1];
w_2_e[0] = w_2_e[1];
A_2[0] = A_2[1];
w_3_e[0] = w_3_e[1];
A_3[0] = A_3[1];
w_4_e[0] = w_4_e[1];
A_4[0] = A_4[1];
acceleration loop_flag = 0;

speedloop_flag += 1;
}
/*************************************************************/
/******************** Data Initializing ******************/
/*************************************************************/

void initial_data(void){

    outportb(PORT+3, 0x0F);
    v=0;
    y=0;
    ref_speed=0;
    AD_1=0;
    AD_1_l=0;
    AD_1_h=0;
    current_1[1]=0;
    current_1[0]=0;
    cur1[1]=0;
    cur1[0]=0;
    voltage_1=0;
    DA_1=0;
    DA_1_l=0;
    DA_1_h=0;
    DA_1_scale=0;
    count_1=0;
    count_1_gap=0;
    pulse_1[1]=0;
    pulse_1[0]=0;
    speed_1[1]=0;
    speed_1[0]=0;

    gasok_1ob[0]=0;

    gasok_1ob[1]=0;

    gasok_1[0]=0;

    gasok_1[1]=0;

    theta_1[1]=0;
    theta_1[0]=0;
    theta_e_1=0;
    theta_ref_1=0;
    w_1_e[1]=0;
    w_1_e[0]=0;
    V_1[1]=0;
    V_1[0]=0;
    A_1[1]=0;
    A_1[0]=0;
    A_1_e[1]=0;
    A_1_e[0]=0;
    AD_2=0;
    AD_2_l=0;
    AD_2_h=0;
    current_2[1]=0;
    current_2[0]=0;
    cur2[1]=0;
    cur2[0]=0;
    voltage_2=0;
    DA_2=0;
    DA_2_l=0;
    DA_2_h=0;
    DA_2_scale=0;
    count_2=0;
    count_2_gap=0;
    pulse_2[1]=0;
    pulse_2[0]=0;
    speed_2[1]=0;
    speed_2[0]=0;

    gasok_2ob[0]=0;

    gasok_2ob[1]=0;

    gasok_2[0]=0;

    gasok_2[1]=0;

    theta_2[1]=0;
    theta_2[0]=0;
    theta_e_2=0;
```

```
theta_ref_2=0;
w_2_e[1]=0;
w_2_e[0]=0;
V_2[1]=0;
V_2[0]=0;
A_2[1]=0;
A_2[0]=0;
A_2_e[1]=0;
A_2_e[0]=0;
AD_3=0;
AD_3_l=0;
AD_3_h=0;
current_3[1]=0;
current_3[0]=0;
cur3[1]=0;
cur3[0]=0;
DA_3=0;
DA_3_l=0;
DA_3_h=0;
DA_3_scale=0;
count_3=0;
count_3_gap=0;
pulse_3[1]=0;
pulse_3[0]=0;
speed_3[1]=0;
speed_3[0]=0;
theta_3[1]=0;
theta_3[0]=0;
theta_e_3=0;
theta_ref_3=0;
w_3_e[1]=0;
w_3_e[0]=0;
V_3[1]=0;
V_3[0]=0;
A_3[1]=0;
A_3[0]=0;
A_3_e[1]=0;
A_3_e[0]=0;
gasok_3ob[0]=0;
gasok_3ob[1]=0;
gasok_3[0]=0;
gasok_3[1]=0;
AD_4=0;
AD_4_l=0;
AD_4_h=0;
current_4[1]=0;
current_4[0]=0;
cur4[1]=0;
cur4[0]=0;
DA_4=0;
DA_4_scale=0;
count_4=0;
count_4_gap=0;
pulse_4[1]=0;
pulse_4[0]=0;
speed_4[1]=0;
speed_4[0]=0;
theta_4[1]=0;
theta_4[0]=0;
theta_e_4=0;
theta_ref_4=0;
w_4_e[1]=0;
w_4_e[0]=0;
V_4[1]=0;
V_4[0]=0;
A_4[1]=0;
A_4[0]=0;
A_4_e[1]=0;
A_4_e[0]=0;
gasok_4ob[0]=0;
gasok_4ob[1]=0;
gasok_4[0]=0;
gasok_4[1]=0;
i=0;
acceleration_loop_flag=0;
speedloop_flag=0;
ref_flag=0;
up_slop = (T_i/3.)*ref;
down_slop = (T_i/4.)*ref;
skip = T_s/T_i;
skip_flag = (int)(skip);
}

/*******************************************************/
void initial_counter(void){
    outportb(OP_PORT, 0x00);
    outportb(TMR_CMD, 0x10);
    outportb(XYCH+8, 0x0F);
    outportb(XYCH+12, 0x0B);
    outportb(XYCH+14, 0x03);
    outportb(XYCH+16, 0x00);
    outportb(XYCH+10, 0x00);
    outportb(ZUCH+8, 0x0F);
    outportb(ZUCH+12, 0x0B);
    outportb(ZUCH+14, 0x03);
    outportb(ZUCH+16, 0x00);
    outportb(ZUCH+10, 0x00);
}

/*******************************************************/
void timer(void){
    outportb(PORT+15, 0x76);
    outportb(PORT+13, 25);
    outportb(PORT+13, 0x00);
    outportb(PORT+15, 0xb6);
    outportb(PORT+14, 20);
    outportb(PORT+14, 0x00);
    outportb(PORT+9, 0xa3);
    oldfunc = getvect(0x0a);
    setvect(0x0a, current);
    outportb(PORT+8, 0x10);
}

// interrupt time set
/*******************************************************/
void control_input_start_end(void){
   DA = 0;
    DA_1_l = (DA&0x00ff) << 4;
    DA_1_h = (DA&0x0ff0) >> 4;
```

```
    DA_2 = DA;
    DA_3_l = (DA&0x00ff) << 4;
    DA_3_h = (DA&0x0ff0) >> 4;
    DA_4 = DA;
    cw_flg = 15;
    outportb(PORT+3, cw_flg);
    outportb(PORT+4, DA_1_l);
    outportb(PORT+5, DA_1_h);
    outportb(PORT+10, DA_2);
    outportb(PORT+6, DA_3_l);
    outportb(PORT+7, DA_3_h);
    outportb(PORT+11, DA_4);
}
/*********************************************************/
void reference1(void){
    ref_rpm += up_slop;
    ref_rpm = (ref_rpm >= ref) ? ref : ref_rpm;
}
void reference2(void){
}
void reference3(void){
    ref_rpm -= down_slop;
    ref_rpm = (ref_rpm <= 0 ) ? 0 : ref_rpm;
}
/*********************************************************/
/*************** realtime Monitoring Main **************/
/*********************************************************/

void main(void){
    char select1;
    clrscr();
    do{
        clrscr();
        selection1();
        select1=getch();
        switch(select1){
            case'1':
                realtime_monitoring();
                break;
            case'2':
                data_saving();
                break;
            case '₩x1b':goto end;
            default : break;
            }
        clrscr();
        }while(1);
    end:;
}
/*********************************************************/
void selection1(void){
  char *ps[ ] = {" [1] Realtime Monitoring ",
    " [2] Ramp, Variable Reference and Data Saving ",
    " [Esc] Quit ",
    " Select job number [1-2] :"};
    clrscr();
    printf("₩n₩n₩n");
    for(i=0;i<=3;i++){
        printf("₩n₩t₩t%s",ps[i]);
        if (i==3) printf("₩n");
    }
}
/*********************************************************/
void realtime_monitoring(void){
    char select2;
    clrscr();
    do{
        clrscr();
        selection2();
        select2=getch();
        switch(select2){
            case'1':
                realtime1();
                break;
            case'2':
                realtime2();
                break;
            case'3':
                realtime3();
                break;
            case'4':
                realtime4();
                break;
            case'5':
                realtime5();
                break;
            case '₩x1b':goto end;
            default : break;
            }
        clrscr();
        }while(1);
    end:;
}
/*********************************************************/
void selection2(void){
    char *ps[ ] = {" [1] No.1 Realtime Monitoring ",
   " [2] No.2 Realtime Monitoring ",
   " [3] No.3 Realtime Monitoring ",
   " [4] No.4 Realtime Monitoring ",
   " [5] All Axe's Position Synchronous Error ",
   " [Esc] Quit ",
   " Select job number [1-2] :"};
    clrscr();
    printf("₩n₩n₩n");
    for(i=0;i<=6;i++){
        printf("₩n₩t₩t%s",ps[i]);
        if (i==6) printf("₩n");
    }
}
/*********************************************************/
void data_saving(void){
    char select3;
    clrscr();
    do{
        clrscr();
        selection3();
        select3=getch();
        switch(select3){
```

```
                case'1':
                        ramp_reference();
                        break;
                case'2':
                        variable_reference();
                        break;
                case '₩x1b':goto end;
                default : break;
                }
            clrscr();
            }while(1);
    end:;
}
/*********************************************************/
void selection3(void){
    char *ps[ ] = {" [1] Ramp Reference and Data Saving
",
    " [2] Variable Reference and Data Saving ",
    " [Esc] Quit ",
    " Select job number [1-2] :"};
    clrscr();
    printf("₩n₩n₩n");
    for(i=0;i<=3;i++){
        printf("₩n₩t₩t%s",ps[i]);
        if (i==3) printf("₩n");
    }
}
/*********************************************************/
void realtime4(void){
    monitor_set();
    initial_data();
    initial_counter();
    monitor_graphset1();
    xx1[0] = xx1[1] = vp.left+90;
    yy1[1] = vp.bottom-349;  /* speed */
    yy2[1] = vp.bottom-229-25; /* current */
    yy3[1] = vp.bottom-129;  /* load */
    yy5[1] = vp.bottom-29-40;  /* synchronous error */
    moveto(xx1[1],yy1[1]);
    control_input_start_end();
    getch();
    outportb(XYCH+8, 0x00);
    outportb(ZUCH+8, 0x00);
    ref_flag = 0;
    timer();
    do{
      if(speedloop_flag==30){
            realtime4_grap();
            speedloop_flag=0;
      }
    }while(!kbhit());
    ref_flag = 2;
    do{
      if(speedloop_flag==30){
            realtime4_grap();
            speedloop_flag=0;
      }
    }while(ref_rpm>0);
    setvect(0x0a, oldfunc);
    control_input_start_end();
    getch();
    getch();
    outportb(XYCH+8, 0x0C);
    outportb(ZUCH+8, 0x0C);
    cleardevice();
    closegraph();
}
/*********************************************************/
void realtime4_grap(void){
    if(x_step>500){
        x_gap += skip;
        monitor_graphset1();
        x_step = 1;
        xx1[0] = xx1[1] =vp.left+90;
        moveto(xx1[1],yy1[1]);
    }
    else{
        x_step++;
        xx1[1] = vp.left+90+(x_step);
        setcolor(1); /*BLUE*/
        setlinestyle(SOLID_LINE,0,NORM_WIDTH);
        y_step = (int)(rpm_4*100./ref_step);
        yy1[1] = (int)(vp.bottom-349-y_step);
        lineto(xx1[1],yy1[1]);
        moveto(xx1[0],yy2[1]);
        setcolor(0); /*BLACK*/
        y_step = (int)(cur4[1]*100./5.);
        yy2[1] = (int)(vp.bottom-229-25-y_step);
        lineto(xx1[1],yy2[1]);
        moveto(xx1[0],yy3[1]);
        setcolor(0); /*BLACK*/
        y_step = (int)((Kt4*cur4[1])*80./1.);
        if(v<10000)
        yy3[1] =  (int)(vp.bottom-130-y_step);
        else if(v<20000)
        yy3[1] = (int)(vp.bottom-130-y_step);
        else
        yy3[1] = (int)(vp.bottom-130-y_step);
        lineto(xx1[1],yy3[1]);
        moveto(xx1[1],yy5[1]);
        setcolor(4); /*RED*/
        y_step = (int)(theta_e_4*120./0.06);
        yy5[1] = (int)(vp.bottom-29-40-y_step);
        lineto(xx1[1],yy5[1]);
```

```
            moveto(xx1[1],yy1[1]);
            xx1[0] = xx1[1];
    }
}
/*******************************************************/
void ramp_reference(void){
    monitor_set();
    initial_data();
    initial_counter();
    control_input_start_end();
    getch();
    outportb(XYCH+8, 0x00);
    outportb(ZUCH+8, 0x00);
    ref_flag = 0;
    timer();
    do{
      if(speedloop_flag==20){
            data_grap();
            speedloop_flag=0;
      }
    }while(ct<350+1);
    ref_flag = 2;
    do{
      if(speedloop_flag==20){
            data_grap();
            speedloop_flag=0;
      }
    }while(ct<500+1);
    setvect(0x0a, oldfunc);
    control_input_start_end();
    outportb(XYCH+8, 0x0C);
    outportb(ZUCH+8, 0x0C);
    data_out();
    cleardevice();
    closegraph();
}
/*******************************************************/
void variable_reference(void){
    monitor_set();
    initial_data();
    initial_counter();
    control_input_start_end();
    getch();
    outportb(XYCH+8, 0x00);
    outportb(ZUCH+8, 0x00);
    ref_flag = 0;
    timer();
    do{
      if(speedloop_flag==20){
            data_grap();
            speedloop_flag=0;
      }
    }while(ct<150+1);
    ref_flag = 2;
    do{
      if(speedloop_flag==20){
            data_grap();
            speedloop_flag=0;
      }
    }while(ct<250+1);
    ref_flag = 1;
    do{
      if(speedloop_flag==20){
            data_grap();
            speedloop_flag=0;
      }
    }while(ct<325+1);
    ref_flag = 0;
    do{
      if(speedloop_flag==20){
            data_grap();
            speedloop_flag=0;
      }
    }while(ct<350+1);
    ref_flag = 1;
    do{
      if(speedloop_flag==20){
            data_grap();
            speedloop_flag=0;
      }
    }while(ct<400+1);
    ref_flag = 2;
    do{
      if(speedloop_flag==20){
            data_grap();
            speedloop_flag=0;
      }
    }while(ct<500+1);
    setvect(0x0a, oldfunc);
    control_input_start_end();
    outportb(XYCH+8, 0x0C);
    outportb(ZUCH+8, 0x0C);
    data_out();
    cleardevice();
    closegraph();
}
/*******************************************************/
void monitor_graphset1(void){
    Fillbox(vp.left,vp.top,vp.right,vp.bottom,15);
    setcolor(0); /*BLACK*/
  rectangle(vp.left+90, vp.top+30, vp.right-49,
  vp.bottom-349);
  rectangle(vp.left+90, vp.top+150, vp.right-49,
  vp.bottom-229);
```

```
 rectangle(vp.left+90, vp.top+270, vp.right-49,
vp.bottom-129);
rectangle(vp.left+90, vp.top+370, vp.right-49, vp.bottom-29);
setcolor(0); /*BLACK*/
settextjustify(CENTER_TEXT,CENTER_TEXT);
settextstyle(0,HORIZ_DIR,1);
x1 = (vp.right-vp.left)/2+20;
y1 = vp.top+10;
outtextxy(x1,y1,"< Four Axes Position Synchronous Control
>");
      setcolor(8); /*DARKGRAY*/
      settextjustify(CENTER_TEXT,CENTER_TEXT);
      settextstyle(0,HORIZ_DIR,1);
      x1 = (vp.right-vp.left)/2+20;
            y1 = vp.bottom-5;
            outtextxy(x1,y1,"Time[s]");
      setcolor(1); /*BLUE*/
      settextjustify(LEFT_TEXT,CENTER_TEXT);
            settextstyle(0,HORIZ_DIR,1);
      x1 = vp.left+92;
            y1 = vp.bottom-454;
      outtextxy(x1,y1,"Speed[rpm]");
            setcolor(0); /*BLACK*/
      settextjustify(LEFT_TEXT,CENTER_TEXT);
      settextstyle(0,HORIZ_DIR,1);
      x1 = vp.left+92;
            y1 = vp.bottom-334;
      outtextxy(x1,y1,"Current[A]");
      setcolor(0); /*BLACK*/
      settextjustify(LEFT_TEXT,CENTER_TEXT);
            settextstyle(0,HORIZ_DIR,1);
      x1 = vp.left+92;
            y1 = vp.bottom-215;
            outtextxy(x1,y1,"Load Disturbance[NM]");
      setcolor(4); /*RED*/
      settextjustify(LEFT_TEXT,CENTER_TEXT);
            settextstyle(0,HORIZ_DIR,1);
      x1 = vp.left+92;
            y1 = vp.bottom-115;
      outtextxy(x1,y1,"Position Synchronous Error[rad]");
      setcolor(1); /*BLUE*/

setlinestyle(USERBIT_LINE,0x01011,NORM_WIDTH);
      settextjustify(RIGHT_TEXT,CENTER_TEXT);
      settextstyle(0,HORIZ_DIR,1);
      x1 = vp.left+90;
      x2 = vp.right-49;
            y1 = y2 = vp.bottom-349;
            gap[0]=(ref_step)/4.;
            for(i=0;i<5;i++){
           gap[1]=gap[0]*i;
           if(i==0||i==4){
             gprintf(x1-10,y1,"%.1f",gap[1]);
             y2 = y1 = y1-25;
           }
           else{
             line(x1,y1,x2,y2);
             gprintf(x1-10,y1,"%.1f",gap[1]);
             y2 = y1 = y1-25;
           }
      }
            setcolor(0); /*BLACK*/
setlinestyle(USERBIT_LINE,0x01011,NORM_WIDTH);
      settextjustify(RIGHT_TEXT,CENTER_TEXT);
      settextstyle(0,HORIZ_DIR,1);
      x1 = vp.left+90;
      x2 = vp.right-49;
            y1 = y2 = vp.bottom-229;
            gap[0] =5./4.;
            for(i=-1;i<4;i++){
           gap[1] = gap[0]*i;
           if(i==-1||i==4){
             gprintf(x1-10,y1,"%.2f",gap[1]);
             y2 = y1 = y1-25;
           }
           else{
             line(x1,y1,x2,y2);
             gprintf(x1-10,y1,"%.2f",gap[1]);
             y2 = y1 = y1-25;
           }
      }
            setcolor(0); /*black*/
setlinestyle(USERBIT_LINE,0x01011,NORM_WIDTH);
      settextjustify(RIGHT_TEXT,CENTER_TEXT);
            settextstyle(0,HORIZ_DIR,1);
      x1 = vp.left+90;
      x2 = vp.right-49;
            y1 = y2 = vp.bottom-129;
            gap[0] = (1.)/4.;
            for(i=0;i<5;i++){
            gap[1] = gap[0]*i;
            if(i==0||i==5){
              gprintf(x1-10,y1,"%.3f",gap[1]);
              y2 = y1 = y1-20;
            }
            else{
              line(x1,y1,x2,y2);
```

```
                gprintf(x1-10,y1,"%.3f",gap[1]);
                y2 = y1 = y1-20;
              }
    }
    setcolor(4); /*RED*/
setlinestyle(USERBIT_LINE,0x01011,NORM_WIDTH);
    settextjustify(RIGHT_TEXT,CENTER_TEXT);
              settextstyle(0,HORIZ_DIR,1);
    x1 = vp.left+90;
    x2 = vp.right-49;
y1 = y2 = vp.bottom-29;
gap[0] = (0.03)/6.;
for(i=-6;i<7;i+=3){
gap[1] = gap[0]*i;
if(i==-6||i==6){
gprintf(x1-10,y1,"%.2f",gap[1]);
y2 = y1 = y1-20;
}
else{
line(x1,y1,x2,y2);
gprintf(x1-10,y1,"%.2f",gap[1]);
y2 = y1 = y1-20;
              }
    }
    setcolor(8); /*DARKGRAY*/
setlinestyle(USERBIT_LINE,0x01011,NORM_WIDTH);
    settextjustify(CENTER_TEXT,CENTER_TEXT);
    settextstyle(0,HORIZ_DIR,1);
    x1 = x2 = vp.left+90+50;
              y1 = vp.top+30;
              y2 = vp.bottom-29;
    gap[0] = (time_sum)/6.6666;
    for(i=1;i<11;i++){
              gap[1] = (x_gap)+gap[0]*i;
              if(i==10){
                gprintf(x1,y2+10,"%.1f",gap[1]);
              }
              else{
                line(x1,y1,x2,y2);
                gprintf(x1,y2+10,"%.1f",gap[1]);
                x2 = x1 = x1+50;
              }
    }
}
/*********************************************************/
void monitor_set(void){
    x_gap = 0;
    x_step = 0, y_step = 0;
    initial_graph();
    setviewport(0,0,MaxX,MaxY,1);
    getviewsettings(&vp);
    setcolor(14);
    settextjustify(CENTER_TEXT,CENTER_TEXT);
    settextstyle(0,HORIZ_DIR,1);
    outtextxy((vp.right-vp.left)/2.,(vp.bottom-vp.top)/2.,
    " Please, Input reference rpm of axes: ");
    scanf("%f",&ref);
    cleardevice();
    initial_time();
}
/*********************************************************/
void initial_time(void){
    time_sum = 20.*T_s*data_sum;
              ref_step = ref*4./3.;
}
/*********************************************************/
void initial_graph(void){
    int GraphDriver;
    int GraphMode;
    int xasp, yasp;
    GraphDriver = DETECT;
    initgraph( &GraphDriver, &GraphMode, "" );
    ErrorCode = graphresult();
    if(ErrorCode != grOk){
              printf("Graphics System Error: %s₩n",
              grapherrormsg( ErrorCode ));
              exit(1);
    }
    getpalette( &palette );
    MaxColors = getmaxcolor()+1;
    MaxX = getmaxx();
    MaxY = getmaxy();
    getaspectratio( &xasp, &yasp );
}
/*********************************************************/
int gprintf( int xloc, int yloc, char *fmt, ... ){
    va_list  argptr;
    char str[250];
    int cnt;
    va_start( argptr, fmt );
    cnt = vsprintf( str, fmt, argptr );
    outtextxy( xloc, yloc, str );
    va_end( argptr );
    return( cnt );
}
/*********************************************************/
void Fillbox(int sx, int sy, int ex, int ey, int color){
    int rec_poly[8];
    setfillstyle(SOLID_FILL, color);
rec_poly[0]=sx,rec_poly[1]=sy,rec_poly[2]=sx,rec_poly[3]=ey
;
rec_poly[4]=ex,rec_poly[5]=ey,rec_poly[6]=ex,rec_poly[7]=sy
;    fillpoly(4, rec_poly);   }
```

참고문헌

(1) Jung-Hoan Byun, Dong-Jun Yeo, "Position Synchronous Control of Two-Axes Driving System by H_∞ Approach", Transactions of KSPE, Vol. 18, No. 2, pp. 192~198, 2001. ,

(2) Seok-Kwon Jeong, Young-Jin Kim, Sam-Sang You, "Precise Position Synchronous Control of Two Axes Rotating Systems by Cooperative Control", Transactions of KSME, Vol. 25, No. 12, pp. 2078~2090, 2001.

(3) Jung-Hoan Byun, Young-Bok Kim, "A Study on Construction of Synchronous Control System for Extension and Stability", Transactions of KSME, Vol. 26, No. 6, pp. 1135~1142, 2002.

(4) Jung-Hoan Byun, "A Study on the Position-Synchronous Control of Coupling Structure by H_∞ Approach", Transactions of KSME, Vol. 26, No. 10, pp. 2052~2059, 2002.

(5) Young-Jin Kim, Bong-Seok Choi, Seok-Kwon Jeong, "High Precision Synchronous Control of Two-Axes Rotating System Using DC Motor and Cooperative Control", Spring annual conference of KSPSE, 2000.

(6) Young-Jin Kim, "High Precision Position Synchronous Control of Multi-Axes Rotating System for DC Motor", PKNU, master's thesis, 2002.

(7) Young-Jin Kim, Seok-Kwon Jeong, Young-Bok Kim, Jung-Hoan Byun, "High Precision 4-Axes Position Synchronous Control by Cross Method", Proceeding of KSPSE, pp. 302~307, 2001.

(8) Bong-Seok Choi, Seok-Kwon Jeong, "Precise Position Synchronous Control of Four Axes Rotating System by Maximum Error Comparison and Acceleration Control", Spring annual conference

of KSME, pp. 129~134, 2003.

(9) Bong-Seok Choi, Seok-Kwon Jeong, "Precise Position Synchronous Control of Four-Axes System Based on Acceleration Control", The 2003 International Symposium on Advanced Engineering, 2003.

(10) Kouhei Ohnishi, Nobutoki Matsui and Yoichi Hori, "Estimation, Identification, and Sensorless Control in Motion Control System", IEEE, Vol. 82, No. 8, pp. 1253~1265, 1994.

(11) Yoichi Hori, "Acceleration Controlled Type Servo System", JIEE, Vol. 108-D, No. 7, pp. 672~677, 1988.

(12) T. KAMANO, T. SUZUKI K. IIDA, Y. KATAOKA, and M. TOMIZUKA, "Multiple Axes Synchronizing System under Adaptive Feedforward Control" , SICE, Vol. 6, No. 11, pp. 488~497, 1993.

(13) V. Utkin, J. Guldner, and J. Shi, "Sliding Mode Control in Electromechanical Systems", 1999.

(14) Robert H. Bishop, "Modern Control System Analysis and Design Using MATLAB", 1993.

(15) 原島文雄, 塚元修巳, "電気制御の基礎", 1986.

(16) 中村正俊, 後藤 聡, 久良修郭, "メカトロサーボ系制御", 1998.

(17) 雨宮好文, 高木章二, "ディジタル制御入門", 1988.

(18) 堀 洋一, "セミナーテキスト, ソフトサーボコントローラの開発と要点と実現"

(19) 金子敏夫, "機械制御工学", 1991.

(20) 石田晴久 訳, "プログラミング言語 C", 1988

(21) 杉本英彦, 小山正人, 玉井伸三, "ACサーボシステムの理論と設計の実際", 1994.

감사의 글

언제나 初心을 잃지 않으려 매진하였지만, 어느새 지나버린 소중한 시간 앞에 미흡한 논문 한편으로 대신하려니 안타깝고 부끄럽기 그지없습니다.

7년전 아무것도 모르는 부족한 저를 제자로 거두어 주셔서 연구자로서의 길을 내딛게 해주시고, 늘 성원해 주시고 돌보아 주시며 학문적 가르침을 아끼지 않으셨던 정석권 지도 교수님께 진심으로 머리 숙여 감사드립니다. 그리고 본 논문심사를 위해 애써주시고 조언해주셨던 제어기계공학과 양주호 교수님, 장지성 교수님께도 진심으로 감사 드립니다. 학부과정에서부터 대학원 과정까지 많은 가르침을 주셨던 김상봉 교수님, 이일영 교수님, 김영복 교수님, 여수대학교 변정환 교수님께도 감사드리며, 많은 격려와 도움을 아끼지 않으셨던 냉동공조공학과 최광환 교수님께도 감사 드립니다.

같이 여러 밤을 새어가며 연구실에서 동거동락한 이진국 선배님, 이제 한국에 적응해서 연구에 매진중인 이 화 선배님, 그리고 묵묵히 열심히 나아가는 동기 김성하 군, 일본에서 새로운 연구를 시작할 정영미 후배님께도 고마움을 전합니다. 그리고 일본 横浜国立大学에서 불철주야 연구에 전념이실 권혁진 선배님, 곧 미국으로 떠날 황동일 선배님, 열심히 직장생활 하고 있는 유성권 선배님, 백정원 선배님, 안혜경 후배님, 곧 신혼살림을 꾸려나갈 동기 김영진 군, 멀리 하와이에서 큰 포부를 다지고 있을 이병혁 군, 그 외 AMCL 연구실의 여러 OB 선·후배님들에게도 감사의 뜻을 전하고 싶습니다. 긍지와 자부심을 늘 갖게 해주셨던 박순식 해군 대위님께도 감사드리며, 멀리 미국에서 새로운 학문을 닦고 있을 친구 이상효 군의 앞날에도 서광이 함께하길 빌고, 자주 커피를 마시며 담소와 정보를 나눴던 김종대 선배님, 정석호 선배님, 그리고 같이 근무하시는 기계공학부 조교 선생님들께도 고마움을 전합니다.

못난 동생과 처남을 위해 애써 주셨던 누님, 매형 내외분께도 깊은 감사의 마음을 전합니다. 끝으로 오늘의 저를 있게 해주시고 학부 및 대학원 전 과정동안 학업에만 전념할 수 있도록 불철주야 사랑과 인내와 희생으로 올바르게 키워주신 부모님께 머리 숙여 깊이 감사 드리며, 이 작은 결실을 바칩니다.

2004年 1月 崔峯碩 拜上