'95 대한전기학회 하계학술대회 논문집(1995. 7. 20~22)

신경 회로망을 이용한 BLDD 모터의 속도 적응 제어기

김 창균, 이 중휘, 윤 명중
한국 과학 기술원 전기및 전자 공학과

Speed Control of BLDD Motor Using Neural Network based Adaptive Controller

Chang-Gyun Kim, Joong-Hui Lee, Myung-Joong Youn
Department of Electrical Engineering, KAIST

Abstract : This Paper presents a novel and systematic approach to a self-learning controller. The proposed controller is built on a neural network consisting of a standard back propagation (BNN) and approximate reasoning (AR). The fuzzy inference and knowledge representation are carried out by the neural network structure and computing, instead of logic inference. An architecture similar to that used by traditional model reference adaptive control system (MRAC) is employed.

1. 서론 (INTRODUCTION)

최근에 이르러 소형 경량화 추세에 의하여 Brushless DC 모터를 이용한 직접 구동 방식이 많이 사용된다. 이 Brushless Direct Drive (BLDD) 모터는 기존에 사용되던 간접 구동 방식에 비하여 그 구조가 간단하여 보수 유지가 간편할 뿐 아니라, 간접 구동 방식에서 문제점으로 지적되던 부분들-Back rush, Sleep, Friction-이 해결되어 정밀 제어가 가능하다는 장점을 갖는다. 반면, BLDD 모터가 갖는 비선형성과 부하의 변화에 대한 민감성은[5] 고전적인 제어 방법으로는 충분한 제어 특성을 얻을 수 없어서 새로운 제어기의 필요성이 강하게 대두된다.

한편 퍼지 논리 제어기를 BLDD에 적용하여 강인성과 천이 상태의 좋은 특성이 점은 많은 연구와 실험을 통하여 증명 되었다 [4]. 퍼지 논리 제어기는 통상 전문가의 경험이나 제어 대상에 관한 지식을 통하여 제어 규칙을 작성하게 된다. 그러나 만일 전문가를 찾을수 없거나 제어 대상에 불확실성이 존재 한다면 그 제어 규칙 작성은 무척 어렵고 힘든 일이 되어 버리고 만다.

이와 같은 문제점을 해결함에 있어서 학습 능력이 있는 신경 회로망과 퍼지 논리 제어기의 결합인 Neuro- Fuzzy 제어기는 우리에게 좋은 해답을 제공한다.[3] 특히 Functional Mapping 방법을 사용한 Nuero -Fuzzy 제어기는 구현이 간편하여 매우 실용적이다. Functional Mapping은 퍼지 논리 제어기 근사 추론과 신경 회로망 학습의 관계를 Functional Mapping 하여 근사 추론 부분을 신경 회로망으로 구현한 방법이다.[1] 그러므로 구현된 신경회로망을 적절히 학습 시킴에 의하여 제어 규칙을 학습 시키는 것과 같은 결과를 얻을 수 있다.

또한 본 논문에서는 신경 회로망의 학습을 위한 Teaching Pattern을 얻기 위하여 Model Reference Adaptive Control (MRAC)에서 사용하는 Adaptation Law와 유사한 방법을 사용한다. 이때 발생된 오차에 의하여 신경 회로망을 바로 Back Propagation 할 수는 없으므로 적절히 수정된 규칙인 Modified Update Rule에 의하여 Back Propagtion을 수행한다.

본 논문에서 제시하는 제어방법이 제어 대상인 BLDD의 상수 및 부하의 변화에 적절히 적용할 수 있는가 하는 것은, 여러 가지 내부적 혹은 외부적인 상태의 변화에 대하여 컴퓨터 모의 실험을 통하여 보여 준다.

2. 시스템 구성 (SYSTEM STRUCTURE)

전체 시스템은 제어 대상인 BLDD 모터와 중심 제어기인 Neuro-Fuzzy Network을 중심으로 다음 그림 1과 같이 구성된다.

2.1 BLDD 모터

BLDD 모터는 Brushless DC 모터를 낮은 속도에서 높은 토크를 발생할 수 있도록 설계한 모터로서 기본적인 구조는 3상 영구 자석형 동기 전동기와 같으므로 d-q 축으로 변환된 전압 방정식으로 모델화 할 수 있고 최대 토크가 출력 되도록 전류제어를 하면 다음과 같이 간략화된 식으로 BLDD 의 동특성을 나타낼 수 있다[5].

$$\frac{d\omega_r}{dt} = \frac{3}{2J}\left(\frac{P}{2}\right)^2 \lambda_m i_{qs} - \frac{B}{J}\omega_r - \frac{P}{2}\frac{T_L}{J} \qquad (2\text{-}1)$$

$$T_e = \frac{3}{2}\left(\frac{P}{2}\right)\lambda_m i_{qs} = k_t i_{qs} \qquad (2\text{-}2)$$

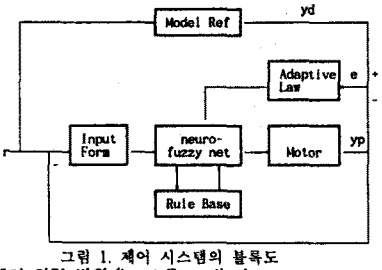


그림 1. 제어 시스템의 블록도

2.2 제어 입력 변환 (Input Formation)

제어기 입력 변환 부분에서는 제어 대상의 출력과 기준을 비교하여 오차값(E)을 얻고 이 오차값의 변화(CE)를 구하여 제어기에 입력하는데 제어기를 마치 고전적인 PI 제어기와 같은 개념으로 생각할 수 있게 한다. 제어기는 E와 CE로부터 제어 입력의 변화(CU)를 Mapping 하는 함수와 같다. 제어기의 Mapping 관계를 F라 하면 다음과 같이 Mapping 관계를 나타낼 수 있고 이 Mapping 은 마치 고전적인 PI 제어기와 같은 역할을 한다[4].

$$F : (E, CE) \rightarrow CU \qquad (2\text{-}3)$$

2.3 Neuro-Fuzzy 제어기

Neuro-Fuzzy 제어기에서는 준비된 제어 규칙에 의하여 오차(E)와 오차의 변화(CE)로부터 제어 입력의 변화(CU)를 근사 추론(AR)하여 제어 입력으로 출력한다. 그러므로 그 기본은 퍼지 논리 제어기와 같으나 그 추론 과정은 논리적인 추론이 아니고 신경 회로망을 이용한 연산에 의한다. 즉 신경 회로망을 이용하여 퍼지 논리의 근사 추론을 모사하는 것이다[1].

2.4 제어 규칙

제어 규칙은 퍼지 논리에 의하여 작성되며 제어기가 제어대상의 상태에 적용하여 짐에 따라 그 제어 규칙이 적절히 변경된다.

본 논문에서 사용한 입출력 퍼지 변수는 PB (Positive Big), PM(Positive Middle), ZE(ZEro), NM (Negative Middle), NB(Negative Big)의 5개 이며 각 퍼지 변수는 삼각 퍼지 숫자로 근사화 하였다. BLDD 모터를 위한 제어규칙은 다음 그림 2와 같이 정리된다[4].

(CE) \ (E)	PB	PM	ZE	NM	NB
PB	PB	PB	PB	PM	ZE
PM	PB	PB	PM	ZE	NM
ZE	PB	PM	ZE	NM	NB
NM	PM	ZE	NM	NB	NB
NB	ZE	NM	NB	NB	NB

그림 2. 제어 규칙

2.5 기준 모델 (Reference Model)

Neuro-Fuzzy 제어기의 Teaching Pattern을 발생 시키기 위하여 기준 모델을 설정하며 기준 모델은 MRAC 에서와 같은 작용을 한다. 본 논문 에서 사용하는 기준 모델은 제어 대상의 특성 및 상수를 안다는 가정에서 설계된 Close Loop System 의 특성과 유사한 선형 모델로 사용한다. 본 논문에서는 다음과 같이 약간의 Over shoot 이 있는 2차 모델을 사용하였다[7].

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \qquad (2\text{-}4)$$

2.6 적응 규칙 (Adaptive law)

기준 모델의 출력과 제어 대상의 출력으로부터 오차를 산출한 뒤 Neuro-Fuzzy 제어기가 학습할 수 있는 적합한 값으로 변환 하여 주는 부분으로 Modified Update Rule 을 사용하며 상세한 설명은 제 4절에서 하도록 한다.

3. NEURO-FUZZY 제어기

3-1. 퍼지 추론

퍼지 논리에서의 추론은 입력과 출력으로부터 Zadeh 의 Composition Rule of Inference 에 의하여 관계 행렬을 작성한 후 입력에 대하여 Max-Min 연산에 의하여 하여 출력을 구하는 방법과 전문가의 경험을 (IF _ THEN _) Rule 로 작성한 뒤 Also 로 각 Rule을 연결하여 추론 규칙으로 하고 입력에 대하여 Mamdani 의 Max-Min Composition Rule을 적용하여 출력을 구하는 방법이 일반적이다. 퍼지 논리 제어에서는 일반적으로 후자의 경우를 많이 사용한다. 만일 L 개의 IF-THEN Rule 이 전문가에 의하여 작성 되었다고 가정하면 먼저 입력 변수 와 출력 변수의 Membership 함수를 정의하고 다음 식에 의하여 적합도 β_i^k 를 구한다.

$$\beta_i^k = MaxMin[C_i(u_i), A_i^k(u_i)] \tag{3-1}$$

여기에서 A_i^k는 제어 규칙의 퍼지 입력 변수이며 C_i는 실제 입력을 퍼지화한 값이다. 위의 식으로부터 구한 적합도 β_i^k에 의하여 해당 출력 변수를 변형한 후 Max Operation 에 의하여 출력 퍼지값 $D(v)$를 얻는다[1].

3.2 신경 회로망의 학습

입력 Layer, Hidden Layer, 출력 Layer 를 포함하는 완전히 연결된 신경 회로망으로 출력은 다음과 같이 구할 수 있다.

$$v_i^o = f_o\left(\sum_{j=1}^{N} w_{ij}^o f_h(s_j) + \theta_i^o\right) \tag{3-2}$$

$$s_j = \sum_{k=1}^{N} w_k^h u_k^o + \theta_j^h \tag{3-3}$$

여기에서 v_i^o와 u_k^o는 입출력 값이며 w_{ij}^o는 각 Neuron 사이의 연결 강도, f_o와 f_h는 출력 및 hidden Layer 에서의 비선형 함수, θ_i^o와 θ_j^h는 각 Neuron의 한계치이다.

Teaching Pattern이 있는 Supervised Learning 에서는 다음과 같은 Learning Rule에 의하여 Neuron 사이의 연결 강도가 학습된다. 연결 강도의 변경 값을 Δw_{ij}라고 하면 다음과 같이 구할 수 있다.

$$\Delta w_{ij} = \eta \delta_j O_i \tag{3-4}$$

$$\delta_j = f'(s_j)(v_j^p - v_j^o) \quad : \text{Output Layer} \tag{3-5}$$

$$\delta_j = f'(s_j)\sum_{k=1}^{N} \delta_k w_{kj} \quad : \text{Hidden Layer} \tag{3-6}$$

위의 식에서 알 수 바와 같이 Teaching Pattern 과 신경 회로망 출력 사이의 오차가 출력 Layer로부터 Hidden Layer 를 거쳐서 입력 Layer 로 전파 되며 Back Propagation 이라고 한다. 이와 같은 Learning Rule에 의하여 점진적으로 신경 회로망이 학습되어 원하는 Mapping 관계를 갖게 된다[6].

3.3 Functional Mapping

퍼지 논리의 근사 추론과 신경 회로망의 Mapping 관점에서 유사성(Isomorphism)을 찾아보고 유사하게 함수적으로 정의하고 구현해 보고자 하는 것이 Neuro- Fuzzy의 Functional Mapping 이다.

먼저 위에서 표현한 신경 회로망을 Mapping 관점에서 살펴본다. 입출력 만을 안다는 것은 알수 없는 어떤 Mapping이 존재한다는 의미를 갖는다. 그러므로 우리가 학습 시키고자 하는 Teaching Pattern 을 Unknown True Mapping ϕ로 정의 할 수 있다.

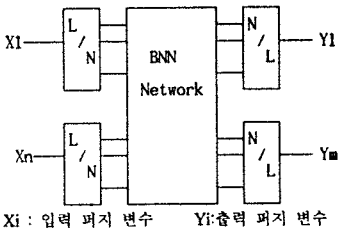
또한 학습되어 있는 신경 회로망은 Unknown True Mapping ϕ를 근사화한 Mapping $\phi^*(u,w)$로서 표현할 수 있으며 입력 u와 연결 강도 w의 함수이다. 그리고 Back Propagation에 의하여 학습한다고 하는 것은 다음 식으로 표현되는 ϕ와 $\phi^*(u,w)$의 Quadratic Error $E(w)$를 최소화 하는 것이다.

$$E(w) = \frac{1}{2}\| \phi(u) - \phi^*(u,w) \|^2 \tag{3-7}$$

다음으로 퍼지 논리의 근사 추론을 Mapping 관점에서 살펴본다. 근사 추론에서 추론 규칙의 작성한다고 하는 것은 입력과 출력만을 아는 Unknown True Mapping Ψ 가 존재한다는 것을 의미하며 단지 입출력이 언어를 표시되는 퍼지 집합으로 된다는 점이 신경 회로망과 다른 점이다. 또한 근사 추론이란 Unknown True Mapping Ψ를 근사적인 Mapping $\Psi(X,W)$으로 표시하는 것이며 변수들의 집합 W의 함수이며 물론 입력 X의 함수이다. 그리고 잘 설정된 근사 추론이란 다음 식으로 표시되는 Ψ와 $\Psi(X,W)$의 Quadratic Error $\widehat{E(W)}$ 를 최소화 한 것이다.

$$\widehat{E(W)} = \frac{1}{2}\| \Psi(X) - \Psi^*(X,W) \|^2 \tag{3-8}$$

위에서 살펴본 바와 같이 신경 회로망과 근사 추론 사이에는 유사성(Isomorphism)이 있으며 단지 신경 회로망은 어느 점(point)에서 다른 점(point)으로의 Mapping인 것에 반하여 근사 추론은 어느 퍼지 집합에서 다른 퍼지 집합으로의 Mapping 인 것이다. 그러므로 퍼지 집합과 점(point) 사이의 변환 장치만 있으면 서로 유사하게 표현할 수 있다. 그리고 이 변환은 이산형의 소속 함수 (Membership Function)를 갖는 퍼지 집합에 대하여 가능하다. 그러므로 이산형 소속 함수에 의한 변환을 삽입하면 다음과 같이 퍼지 논리의 근사 추론을 신경 회로 망으로 구현할 수 있다[1].



Xi : 입력 퍼지 변수 Yi:출력 퍼지 변수
L/N : Ling./Num. 변환

그림 3 신경 회로망에 의한 근사 추론

4. ADAPTIVE MECHANISM

Neuro-Fuzzy Network 을 학습 시키기 위하여는 기준 모델과 제어 대상의 오차로부터 Network의 출력 Layer 의 오차를 구하여야 하는데 그 방법을 Modified Update Rule 이라고 한다.

퍼지 논리의 근사 추론을 Functional Mapping한 신경 회로망을 제어기로 사용할 경우 다음 식에 의하여 제어 입력값 u를 구한다.

$$u = h(M_j v_j) = \frac{\sum_{j=1}^{N} M_j v_j}{\sum_{j=1}^{N} v_j} \tag{4-1}$$

v_j : j 번째 node 의 출력, M_j : 출력 node 의 양자화 값

Neural Network 에서의 Update Rule 은 다음 식과 같이 기준 모델 출력 y_d와 제어 대상 출력 y_p의 오차를 취하여 Quadratic $E(t)$를 구한 뒤 Gradient Discent Rule 을 적용하여 얻어진다.

$$E(t) = \frac{1}{2}\| y_d(t) - y_p(t) \|^2 = \frac{1}{2}(y_d - y_p)^2 \tag{4-2}$$

Hidden Layer 와 출력 Layer 사이의 연결 강도 w_{ij}의 Update 값 Δw_{ij}은 Gradient Discent Rule 에 의하여 다음과 같이 표시된다.

$$\Delta w_{ji} = -\rho \frac{\partial E}{\partial w_{ji}} = -\rho \frac{\partial E}{\partial v_j}\frac{\partial v_j}{\partial w_{ij}} = -\rho \frac{\partial E}{\partial v_j} O_i^h \tag{4-3}$$

$$\frac{\partial E}{\partial v_j} = \frac{\partial E}{\partial y_p}\frac{\partial y_p}{\partial u}\frac{\partial u}{\partial v_j} = -(y_d - y_p)\frac{\partial y_p}{\partial u}\frac{\partial u}{\partial v_j} \tag{4-4}$$

$$\frac{\partial u}{\partial v_j} = h'(M_j v_j) = \frac{M_j - 1}{(\sum_{j=1}^{N} v_j)^2} \tag{4-5}$$

forward jacobian : $g = \frac{\partial y_p}{\partial u}$

그러므로 w_{ij}의 Update 값 Δw_{ij}은 다음과 같이 정리된다.

$$\begin{aligned}\Delta w_{ji} &= \rho[(y_d - y_p)\frac{\partial y_p}{\partial u}\frac{\partial u}{\partial v_j}] O_i^h \\ &= \rho[(y_d - y_p)\, g\, h'(M_j v_j)] O_i^h\end{aligned} \tag{4-6}$$

본 논문에서는 내부 상수와 부하가 불확실 한 상태를 점진적으로 적응하는 것을 목표로 하므로 g의 값을 알 수는 없다. 그러나 일반적으로 g의 부호는 알 수 있다고 가정하여도 큰 무리는 아니므로 위의 식에 sgn[g]만을 반영하고 g의 절대 값은 Learning Rate ρ에 흡수시켜 다음과 같은 Modified Update Rule을 완성 시킨다.

$$\Delta w_{ji} = \rho\, sgn[g]\, (y_d - y_p)\, h'(M_j v_j)\, O_i^h \tag{4-7}$$

5. 컴퓨터 모의 실험 결과

컴퓨터 모의 실험에서 사용한 모터의 상수는 다음과 같다.

Kt = 0.512 [Nm/A] J = 0.392 [Nm sec^2]

B = 2.5 [Nm sec] P = 8 [pole]

상수의 불확실성을 고려하여 J 는 Nominal Value 와 Nominal Value의 2배에 대하여 실험하였으며 B 도 Nominal Value 와 Nominal Value의 3배에 대하여 실험하였다.

실험에서 사용한 기준 모델은 Nominal Plant에서 조정된 Neuro-Fuzzy 제어기를 적용 하였을 때의 특성과 유사한 Model 1 과 Nominal Plant 와 Neuro-Fuzzy 제어기를 결합한 Model 2로 하였다. 각 Model을 정리하여 적으면 다음과 같다.

Model 1 : ζ = 0.83 ω_n = 3.5[rad/sec]

Model 2 : Nominal System

그림 4와 그림 5 에서는 Model 1을 사용하고 제어대상은 Nominal 상태일 때 Sin 함수의 속도 변화 명령을 추적하는 추적 제어 상태를 보인 것이다. t=2.0sec 에서 적응제어를 시작한 경우를 그림 4 에서 보여주며 초기에는 큰 오차를 갖으면서 기준 모델의 출력에 접근 하지만 그림 5 에서 보는 바와 같이 시간이 경과하면 기준 모델의 출력과 거의 유사한 상태로 됨을 알 수 있으며 지속적인 학습을 통하여 점점 그 오차는 작이 진다.

그림 6과 그림 7 에서는 제어 대상의 상수인 J와 B가 부하의 변

동에 의하여 변경 되었을 때의 특성을 보여준다. 그림 6 에서는 상수가 변경된 경우 Neuro-Fuzzy 제어기 만으로는 명령에 대하여 잘 추적하지 못함을 보여준다. 그림 7 은 50 sec 가량 학습후의 상태를 보여 주는데 결국은 기준 모델의 출력과 유사해 짐을 알 수 있다.

그림 4 - 그림 7 에서 알 수 있는 것은 Neuro- Fuzzy 제어기가 학습 함에 따라 제어 대상의 상수에 관계없이 결국은 기준 모델의 특성과 같아지며 Nominal 상태와 다른 정도에 따라 필요한 학습 시간만이 다르다는 것을 알수 있다.

그런데 위의 실험에서 Neuro-Fuzzy 제어기 만을 Nominal한 제어 대상에 적용한 경우 추적 오차가 "0"으로 접근함에 반하여 기준 모델로 학습 시킨 경우 그만큼의 정밀도를 갖으려면 많은 학습 시간이 필요함을 알 수 있다. 즉 초기 Neuro-Fuzzy제어기는 비선형의 퍼지 논리 제어기를 학습 하였으므로 선형의 기준 모델을 다시 학습 시키기 위하여는 많은 시간이 필요할 것이고 그 성능은 퍼지 논리 제어기가 갖었던 좋은 특성은 잃어 버릴 수 밖에 없다. 그러므로 기준 모델을 Model 2 즉 Nominal 상태에서의 Neuro-Fuzzy 제어기가 적용된 시스템으로 하는 것이 더 자연스럽다.

그림 8과 그림 9 에서는 Model 2 을 적용한 경우의 특성을 보여준다. 학습은 t=4.0sec 에서 시작 하였다. 이 그림에서 보면 점진적으로 기준 모델과 같아질 뿐 아니라 완전히 같아짐을 알수 있다. 즉 비선형 퍼지 논리의 근사 추론을 학습한 Neuro-Fuzzy 제어기는 같은 정도의 비선형성을 갖는 모델을 기준 모델로 하는 것이 바람직 하다.

6. 결론

본 논문에서는 MRAC 형태의 Nuero-Fuzzy 제어기를 구현함에 있어서 Functional Mapping에 의하여 퍼지 논리 제어기의 근사 추론을 신경 회로망으로 모사 하였으며 기준 모델을 학습하는 방법은 새롭게 정의된 Modified Update Rule 을 사용하였다. 제어 대상은 부하의 변동에 민감한 BLDD 모터에 적용 하여 관성과 마찰이 변경된 경우에 적용되어 가는 것을 컴퓨터 모의 실험을 통하여 보였다. 여기에서 어떤 기준 모델을 설정하여도 결국은 Neuro-Fuzzy 제어기가 잘 학습함을 보였고 같은 형태의 비선형성을 갖는 기준 모델에서 효과적으로 적용되어 감을 보였다. 그러므로 본 논문에서 제안하는 제어 방법은 기존의 퍼지 논리 제어기의 단점인 적용성 문제를 해결할 수 있는 효과적인 방법이며 더불어 퍼지 논리 제어기의 장점도 포함한다. 더 나아가 본 논문에서 제안하는 제어 방법에 의하면 정확하게 조정되지 않은 Neuro-Fuzzy 제어기를 적절하게 조정할 수 있는 Teaching Pattern 도 제공할 수 있다. 본 논문에서 제안하는 제어기는 퍼지 논리 제어기의 근사 추론 부분만을 학습을 통하여 조정하는데 통상 퍼지 논리 제어기 설계시 근사 추론 부위도 조정하지만 양자화시 및 출력에 대한 Scaling Factor 도 조정하므로 이 Scaling Factor 도 같이 학습 할 수 있는 방법을 추가하면 제어기의 완성도를 높일 수 있다.

7. 참고 문헌 (REFERENCE)

[1] Junhong Nie and D. A. Linkens, "Neural network based approximate reasoning: principle and inplementation", *International Journal of Control*, vol. 60, no. 2, pp. 399-411, 1992.

[2] T. J. Procyk and E. H. Mamdani, "A linguistic self organizing process controller", *automatica*, no. 15, pp. 15-30, 1979.

[3] H. Takaki and A. Hayashi, "NN-driven fuzzy reasoning", *International Journal of approxinate reasoning*, no. 5, pp. 191-212, 1991.

[4] J. S. Ko, G. J. Hwang and M. J. Youn "Robust position control of BLDD Motor using integral proposition plus fuzzy logic control", *IECON*, vol. 1, pp. 213-218, 1993.

[5] P. C. Krause, *"Analysis of electric machinary"*, McGraw-Hill, 1984

[6] S. Haykin, *"Neural networks - A comprehensive foundation"* Macmillan, 1994

[7] K. S. Narendra and A. M. Anaswany, *"Stable adaptive system"*, Prantice Hall, 1989.

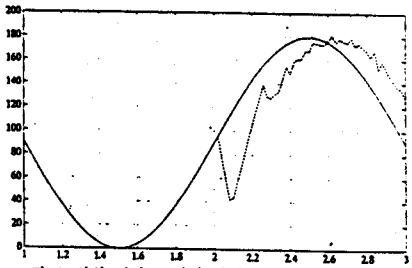


그림 4. 실험 결과 - 천이 상태 - (Model 1, Nominal Plant)

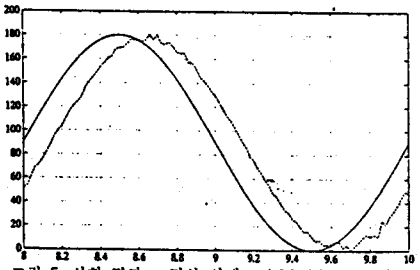


그림 5. 실험 결과 - 정상 상태 - (Model 1, Nominal Plant)

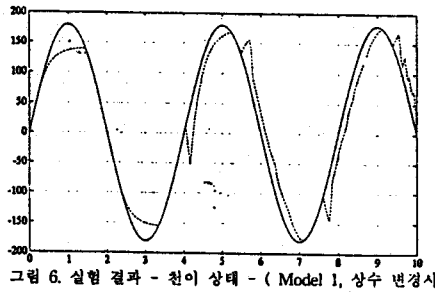


그림 6. 실험 결과 - 천이 상태 - (Model 1, 상수 변경시)

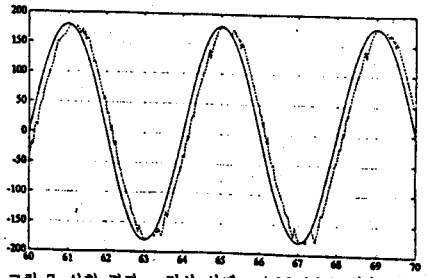


그림 7. 실험 결과 - 정상 상태 - (Model 1, 상수 변경시)

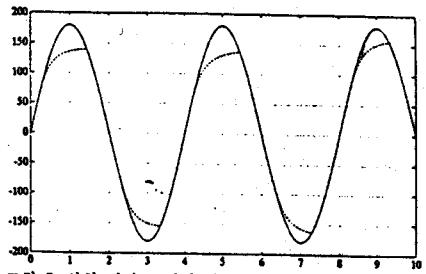


그림 8. 실험 결과 - 천이 상태 - (Model 2, 상수 변경시)

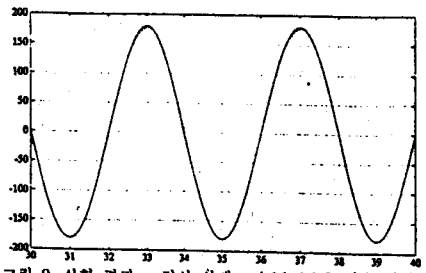


그림 9. 실험 결과 - 정상 상태 - (Model 2, 상수 변경시)